ПРОЕКТУВАННЯ ПРОГРАМНИХ ДОДАНКІВ

ЧАСТИНА І
КОМП'ЮТЕРНІ ПРАКТИКУМИ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ПРОЕКТУВАННЯ ПРОГРАМНИХ ДОДАНКІВ

Частина І

КОМП'ЮТЕРНІ ПРАКТИКУМИ

*Рекомендовано Методичною радою НТУУ «КПІ ім. Ігоря Сікорського»
як навчальний посібник для студентів,
які навчаються за спеціальністю 151 «Автоматизація та комп'ютерно-інтегровані технології»*

Київ
КПІ ім. Ігоря Сікорського
2018

ПРОЕКТУВАННЯ ПРОГРАМНИХ ДОДАНКІВ: ЧАСТИНА І. КОМП'ЮТЕРНІ ПРАКТИКУМИ [Електронний ресурс]: навч. посіб. для студ. спеціальності 151 – «Автоматизація та комп'ютерно-інтегровані технології» / КПІ ім. Ігоря Сікорського; уклад.: В. І. Бендюг, Б. М. Комариста. – Електронні текстові данні (1 файл: 4,13 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2018. – 285 с.

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол № 10 від 21.06.2018р.)

Електронне мережне навчальне видання

ПРОЕКТУВАННЯ ПРОГРАМНИХ ДОДАНКІВ

ЧАСТИНА І

КОМП'ЮТЕРНІ ПРАКТИКУМИ

Укладачі: Бендюг Владислав Іванович, канд. техн. наук, доцент
Комариста Богдана Миколаївна, канд. техн. наук, ст. викл.

Відповідальний редактор О. О. Квітка, канд. хім. наук, доцент

Рецензенти: Прізвище, ініціали, науковий ступінь, вчене звання
Прізвище, ініціали, науковий ступінь, вчене звання

Зміст

Вступ 8
Комп'ютерний практикум №1 10
Робота з масивами в консольному додатку CLR 10
Теоретичні відомості 11
1.1 Створення консольного додатку CLR 11
1.2 Специфіка C++/CLI: форматування виведення 12
1.3 Клавіатурне введення в C++/CLI 15
1.4 Об'єкти класу String 17
1.5 Об'єднання рядків 18
1.6 Зміна рядків 19
1.7 Порівняння рядків 22
1.8 Пошук рядків 22
1.9 Масиви середовища CLR 24
1.10 Сортування одномірних масивів 29
1.11 Пошук в одномірному масиві 31
1.12 Багатомірні масиви 35
1.13 Зубчасті масиви 36
Приклад програмної реалізації 43
Приклад створення консольного додатку CLR. 43
Програмний код 45
Завдання до комп'ютерного практикуму №1 48
Контрольні питання 50
Комп'ютерний практикум №2 51
Створення додатку Windows Forms 51
Теоретичні відомості 52
2.6 Технологія Windows Forms 52
2.7 Створення нового додатку Windows Forms 53
2.8 Стандартні простори імен бібліотеки .NET 60
2.9 Зміна властивостей форми 63
2.10 Запуск додатку 65
Приклад програмної реалізації 66
Приклад створення проекту Windows Forms з використанням класів Form, GroupBox, Panel, Lable, TextBox, Button, RichTextBox. 66
Приклад створення проекту Windows Forms з використанням класів Form, ComboBox, ListBox, Button, RichTextBox. 70
Програмний код 77

Завдання до комп'ютерного практикуму №2 88

Контрольні питання 92

Комп'ютерний практикум №3 93

Створення меню та панелей інструментів 93

Теоретичні відомості 94

3.1 Додавання головного меню 94

3.2 Додавання обробників подій до пунктів меню 97

3.3 Створення контекстного меню 99

3.4 Додавання панелі інструментів 101

Приклад програмної реалізації 104

Приклад створення проекту Windows Forms з використанням класів MenuStrip, ContextMenuStrip та ToolStrip. 104

Програмний код 113

Завдання до комп'ютерного практикуму №3 123

Контрольні питання 129

Комп'ютерний практикум №4 130

Табличне введення даних 130

Теоретичні відомості 131

4.1 Введення табличних даних 131

Приклад програмної реалізації 131

Приклад створення проекту Windows Forms для відображення даних у табличному вигляді та збереженя даних у файлі XML. 131

4.2 Побудова графіку з використанням елементу Chart 134

Приклад програмної реалізації 135

Приклад використання елементів керування Chart і DataGridView, виведення графіку (діаграми) залежнсті обсягів продажів від часу за місяцями. 135

Програмний код 139

Завдання до комп'ютерного практикуму №4 147

Контрольні питання 152

Комп'ютерний практикум 5 153

Редагування графічних даних 153

Теоретичні відомості 154

5.1 Найпростіше виведення графічного зображення у форму 154

Приклад програмної реалізації 157

Приклад виведення графічного зображення з вказаного файлу за допомогою об'єкту класу Image та методу малювання зображення у формі DrawImage графічного об'єкту Graphics з аргументу е процедури OnPaint. 157

Приклад виведення зображення з файлу за натисканням кнопки, шляхом створення об'єктів класу Image та Graphics. 158

5.2 Використання елементу PictureBox для відображення растрового файлу з можливістю прокрутки 159

Приклад програмної реалізації 160

Приклад відкриття довільного графічного файлу з можливістю прокрутки зображення та зберігання його під іншим ім'ям за допомогою об'єктів класу OpenFileDialog та SaveFileDialog. 160

5.3 Побудова графіку методами класу Graphics 162

Приклад програмної реалізації 162

Приклад побудови графіку в ході виконання програми за допомогою об'єктів класів Bitmap, Graphics і Pen та методів DrawString, DrawLine та DrawEllipse класу Graphics. 162

5.4 Створення багатосторінкового інтерфейсу 166

Приклад програмної реалізації 168

Приклад відкриття різних типів файлів з використанням діалогового вікна OpenFileDialog на різних сторінках TabPage елементу TabControl. 168

Програмний код 172

Завдання до комп'ютерного практикуму №5 194

Контрольні питання 200

Комп'ютерний практикум 6 201

Робота з базами даних 201

Теоретичні відомості 202

6.1 Технологія ADO.NET 202

6.2 Створення бази даних MS Access 203

6.3 Запис структури таблиці в порожню базу даних MS Access. Програмне підключення до БД 207

Приклад програмної реалізації 207

Приклад програмного підключення до бази даних за допомогою об'єкту класу OleDbConnection з задаванням SQL-запиту в об'єкті класу OleDbCommand і виконанням запиту функцією ExecuteNonQuery(). 207

6.4 Додавання записів в таблицю бази даних MS Access 209

6.5 Зчитування всіх записів з таблиці бази даних за допомогою об'єктів класів Command, DataReader та елементу керування DataGridView 214

Приклад програмної реалізації 214

Приклад виведення існуючої таблиці БД в об'єкт класу DataGridView за допомогою об'єктів класів Command та DataReader. 214

6.6 Зчитування даних з БД в сітку даних DataGridView з використанням об'єктів класів Command, Adapter та DataSet 215

Приклад програмної реалізації 216

Приклад виведення існуючої таблиці БД в об'єкт класу DataGridView за допомогою об'єктів класів Command, Adapter та DataSet. 216

6.7 Оновлення записів в таблиці бази даних MS Access 217

Приклад програмної реалізації 218
Приклад модифікації записів БД. 218
6.8 Видалення записів з таблиці бази даних з використанням SQL-запиту і об'єкта класу Command 220
Програмний код 223
Завдання до комп'ютерного практикуму №6 228
Контрольні питання 229
Комп'ютерний практикум №7 230
Використання функцій зовнішніх програм 230
Теоретичні відомості 231
7.1 Перевірка правопису засобами MS Word 231
Приклад програмної реалізації 234
Приклад створення додатку з перевіркою правопису засобами MS Word. 234
Приклад програмної реалізації 237
Приклад створення таблиці в MS Word. 237
7.2 Використання функцій MS Excel 240
7.3 Рішення системи рівнянь за допомогою функцій MS Excel 244
Приклад програмної реалізації 245
Приклад рішення системи лінійних рівнянь з використанням функцій MS Excel. 245
Приклад програмної реалізації 247
Приклад побудови діаграм засобами MS Excel 247
Програмний код 258
Завдання до комп'ютерного практикуму №7 278
Контрольні питання 280
Рекомендована література 281
Додаток А 283
Формули для довідок 283

Вступ

Згідно робочого навчального плану кредитний модуль «Прикладне програмне забезпечення - 3. Проектування програмних доданків» дисципліни «Прикладне програмне забезпечення» викладається студентам четвертого року підготовки ОКР «бакалавр» спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» у восьмому навчальному семестрі. Матеріал кредитного модуля базується на дисциплінах «Комп'ютерні технології та програмування - 1. Основи алгоритмізації», «Комп'ютерні технології та програмування - 2. Програмування типових задач», «Комп'ютерна техніка та організація обчислювальних робіт», «Комп'ютерні технології та програмування - 3. Розробка інтерфейсу користувача», «Інформаційні системи та комплекси», «Прикладне програмне забезпечення - 2. Технології об'єктно-орієнтованого програмування». Знання уміння та навички, одержані під час вивчення даного модуля, у подальшому використовуються в усіх дисциплінах, які потребують програмної реалізації розрахунків на комп'ютері, і в першу чергу – в курсах «Ідентифікація та моделювання об'єктів автоматизації», «Основи проектування систем автоматизації і систем керування експериментом», «Методи штучного інтелекту та їх застосування в хімічній технології», в курсових і дипломних роботах і проектах.

Метою навчальної дисципліни є формування у студентів здібностей:

- ПК-2 - Здатність розробляти, проектувати та вдосконалювати елементи комп’ютерно-інтегрованих систем управління об’єктами і процесами, алгоритми їх функціонування.
- ПК-3 - Здатність використовувати сучасні методи і засоби проектування в розробці алгоритмічного та програмного забезпечення систем автоматизації та комп’ютерно-інтегрованих технологій.
- ПК-5 - Здатність розробляти, налагоджувати та вдосконалювати програмне забезпечення комп’ютерно-інтегрованих систем.
- ПК-11 - Здатність застосовувати комп’ютерну техніку та розробляти прикладні програмні продукти для вирішення виробничо-технічних задач.
- ПК-12 - Здатність розробляти програмно-алгоритмічні засоби реалізації методів управління в автоматизованих системах та комп’ютерно-інтегрованих технологіях.

Згідно з вимогами освітньо-професійної програми студенти після засвоєння навчальної дисципліни мають продемонструвати такі результати навчання:

знання:

- стандартів сучасних об’єктно-орієнтованих мови програмування С++;
- Інтегрованого середовища розробки програмних додатків Microsoft Visual Studio;
- сучасних методів конструювання програм засобами об'єктно-орієнтованої мови програмування С++;
- ефективних методів збереження і обробки інформації;

уміння:

- самостійно розробляти ефективні алгоритми і програми на сучасній алгоритмічній мові для вирішення поставленої задачі;
- налагоджувати програми на ПК до надійної працездатності;
- створювати додатки з використанням технології Windows Form;

досвід:

- виконувати налаштування середовища програмування для ефективної роботи;
- оформляти розроблені програми згідно вимогам ЄСПД.

Комп'ютерний практикум №1

Робота з масивами в консольному додатку CLR

Мета: вивчити прийоми створення нового консольного додатку CLR в Visual C++. Засвоїти методи *виведення та введення* даних в консольному додатку CLR. Відпрацювати принципи роботи з рядковими даними. Вивчити засоби управління *форматом виведення* даних в консольних додатках CLR. Ознайомитись з *одномірними та багатомірними масивами* мови C++/CLI. Відпрацювати навички *пошуку значень* в масивах C++/CLI. Ознайомитись з прийомами *сортування* даних в масивах C++/CLI.

Завдання: створити новий консольний додаток CLR, додати в нього необхідний програмний код для створення двомірного масиву, введення необхідних значень та виведення масиву на екран після реалізації поставленої задачі згідно отриманого варіанту завдання.

Загальні вимоги.

1) Створити консольний додаток CLR.
2) Передбачити введення всіх необхідних даних, в тому числі розмірності масиву.
3) Програмний код має бути чітко структурований.
4) Імена об'єктів мають нести сенсові навантаження.
5) Програмний код має супроводжуватись коментарями в тексті програми.
6) Організувати структуроване форматоване виведення масиву в консольному додатку.

Вимоги до виконання.

1) Розмірність масиву вводити через запит до користувача.
2) Масив значень зберігати у створеному масиві C++/CLI з рангом 2.
3) Значення елементів масиву задавати за допомогою генератора випадкових чисел чи з зовнішнього файлу.
4) Значення масиву вивести в консольному вікні у форматованому та чітко структурованому вигляді.
5) Всі знайдені згідно завдання значення вивести на екран у форматованому вигляді з поясненнями.

Теоретичні відомості

1.1 Створення консольного додатку CLR

Створіть новий проект і виберіть в якості типу проекту CLR, а як шаблон - CLR Console Application (Консольне додаток CLR). Потім введіть ім'я проекту Прізвище_КПР1, як показано на рис. 1.1.

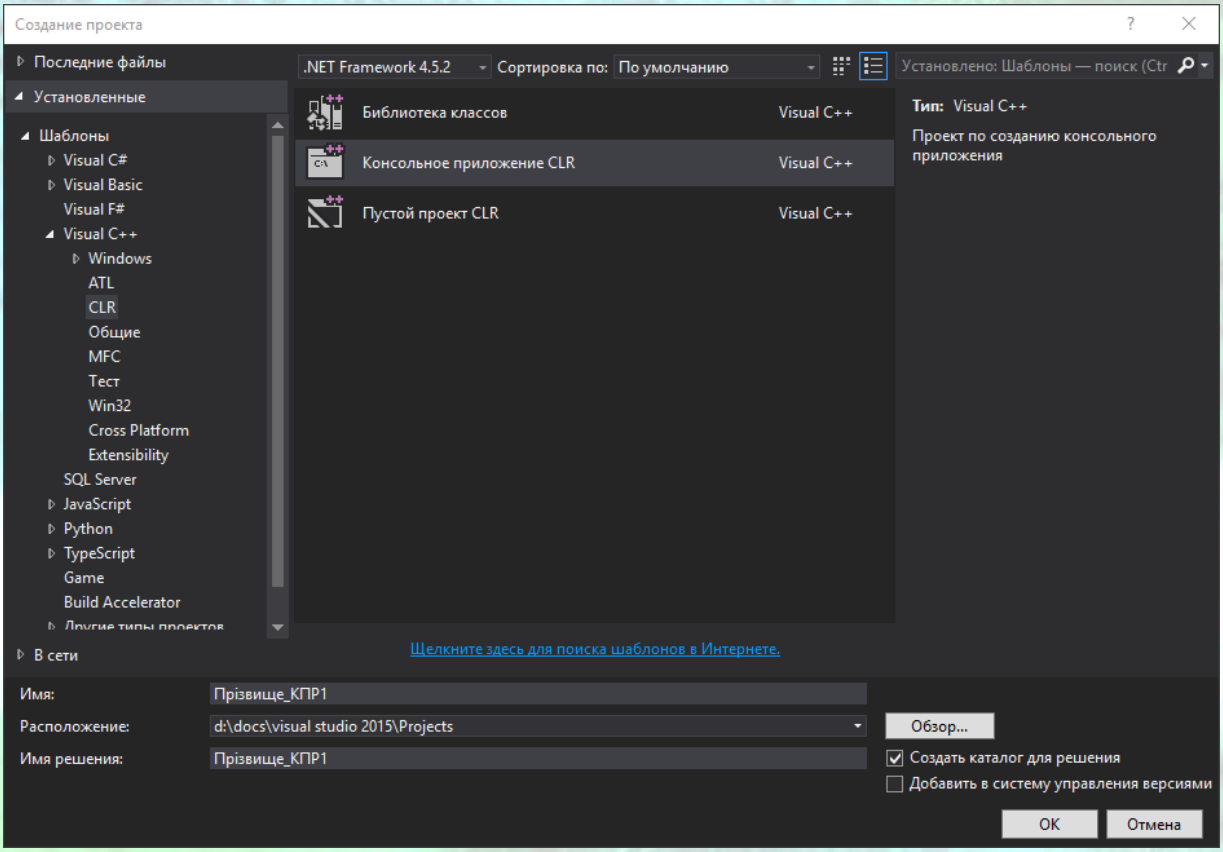

Рисунок 1.1 – Вікно створення консольного додатку CLR

Після клацання на кнопці OK майстер додатків Application Wizard створює проект, що містить наступний код.

```
// Прізвище_КПР1.cpp: главный файл проекта.

#include "stdafx.h"

using namespace System;

int main(array<System::String ^> ^args)
{
    Console::WriteLine(L"Здравствуй, мир!");
    return 0;
}
```

Функція WriteLine() - це функція C++/CLI, яка визначена в класі Console простору імен System. Клас Console надає стандартні потоки

введення і виведення, відповідні клавіатурі і командному рядку консольного вікна. Іншими словами, функція `WriteLine()` пише все, що знаходиться між дужками, наступними за її ім'ям, в командний рядок, додаючи при цьому символ нового рядка, щоб перемістити курсор в початок наступного рядка екрану. Літера `L` перед рядком вказує на те, що це рядок, який складається з «широких» символів, кожен з яких займає 2 байта.

Також в C++/CLI існує *функція `Write()`* класу `Console` - це, по суті, те ж саме, що функція `WriteLine()`, з єдиною відмінністю - вона не додає до зазначеного виводуу символ нового рядка. Тому використовуйте функцію `Write()`, коли потрібно вивести два або більше елементів даних в один рядок декількома окремими операторами.

Значення, які ви ставите в дужках, що слідують за ім'ям функції, називаються *аргументами*. Залежно від того, як написана функція, вона може отримувати при виклику один, кілька аргументів або взагалі не отримувати аргументів. Якщо потрібно передати більше одного аргументу, розділяйте їх комами. Функції виводу класу `Console` заслуговують більш детального розгляду, тому опишемо їх докладно.

1.2 Специфіка C++/CLI: форматування виведення

Обидві функції - і `Console::Write()`, і `Console::WriteLine()` - мають засоби управління форматом виведення, і цей механізм працює абсолютно однаково в обох функціях. Найпростіше зрозуміти його на прикладі. Для початку подивимося, як можна за допомогою одного оператора отримати виведення, яке в попередньому прикладі формувався трьома операторами.

```
int packageCount = 25;
Console::WriteLine(L"Наявні {0} пакетів.", packageCount);
```

Тут другий оператор забезпечить таке ж виведення, як ви бачили в попередньому прикладі. Перший аргумент функції `Console::WriteLine()` - це рядок `L"Наявні {0} пакетів."`, в якому фрагмент `{0}` позначає місце, куди буде поміщений другий аргумент. Цей фрагмент містить в собі форматний рядок, який застосовується для виведення другого аргументу, хоча в даному випадку він гранично простий і складається з одного нуля. Аргументи, які прямують у функції `Console::WriteLine()` за першим, пронумеровані по порядку, починаючи з нуля.

```
// Порядок аргументів: 0 1 2 і т.д.
Console::WriteLine("Рядок формату", arg2, arg3, arg4, ...);
```

Таким чином, нуль, укладений у фігурні дужки в попередньому фрагменті коду, вказує на те, що аргумент `packageCount` повинен замінити місце фрагмента `{0}` в рядковому аргументі при виведенні його на консоль.

Якщо захочете вивести ще й вагу пакетів разом з кількістю, напишіть так.

```
int packageCount = 25; double packageWeight = 7.5;
Console::WriteLine(L"Наявні {0} пакетів вагою {1} кілограм.", packageCount, packageWeight);
```

Тепер оператор виведення має три аргументи, і до другого і до третього в форматному рядку виконується звернення за номерами `0` і `1` відповідно. Тому це дасть наступне виведення.

```
Наявні 25 пакетів вагою 7,5 кілограм.
```

Можете також написати оператор, який виводить два останніх аргументу в зворотному порядку:

```
Console::WriteLine(L"Наявні {1} пакетів вагою {0} кілограм.", packageWeight, packageCount);
```

Тепер `0` в рядку формату відноситься до змінної `packageWeight`, а `1` - до `packageCount`; виведення буде точно такимо ж, як раніше.

Ви також можете вказати, як будуть представлені дані в командному рядку. Припустимо, потрібно вивести значення з плаваючою комою `packageWeight` з двома десятковими розрядами після точки. Це можна зробити наступним чином:

```
Console::WriteLine(L"Наявні {0} пакетів вагою {1:F2} кілограм.", packageCount, packageWeight);
```

У підрядку `{1:F2}` двокрапка відокремлює значення індексу `1`, що ідентифікує аргумент, від наступного за ним вказівника формату `F2`. Літера `F` в визначенні формату вказує, що виведення повинне бути в формі «`±ddd.dd ...`» (де `d` представляє десяткову цифру, а `2` - це кількість розрядів після точки). Виведення цього оператора буде таким:

```
Наявні 25 пакетів вагою 7,50 кілограм.
```

У загальному випадку, можете використовувати специфікацію формату у вигляді `{n,w:Axx}`, де `n` - значення індексу, що вказує номер аргументу, наступного за форматованим рядком; `w` - необов'язкова специфікація ширини поля; `A` - односимвольна специфікація формату значення; `xx` - необов'язкове одно- або двозначне число, що задає точність виведення значення. Специфікація ширини поля - ціле зі знаком. Значення буде вирівняно вправо в полі `w`, якщо ширина позитивна, і вліво - якщо негативна. Якщо значення займає менше

знаків, ніж зазначено в аргументі w, то виведення доповнюється пробілами; якщо значення не вміщується в ширину w, то специфікація ширини ігнорується. Ось ще один приклад.

```
Console::WriteLine(L"Пакетів:{0,3} Вага:{1,10:F3} кілограм.", packageCount, packageWeight);
```

Кількість пакетів виводиться в поле шириною три знака, а вага - в поле шириною десять знаків, тому в результаті отримаємо:

```
Пакетів: 25 Вага:     7,500 кілограм.
```

Існують і інші специфікатори формату, які дозволяють представляти різні типи даних різноманітними способами. Нижче наведені деякі з найбільш часто використовуваних специфікаторів (табл. 1.1).

Таблиця 1.1 – Основні специфікатори форматування консольного виведення C++/CLI

Специфікатор формату	Опис
C або c	Виводить значення в грошовому форматі
D або d	Виводить ціле число в десятковому вигляді. Якщо вказати більш високу точність десяткових знаків в числі, то воно буде доповнено нулями зліва
E або e	Виводить значення з плаваючою комою в науковій нотації, тобто з експонентою. Значення точності вказує кількість десяткових розрядів після точки
F або f	Виводить значення з плаваючою комою як число з фіксованою комою в вигляді ±ddd.dd...
G або g	Виводить значення в найбільш компактному вигляді, в залежності від його типу і зазначеної точності. Якщо точність не вказана, приймається значення точності, задане за замовчуванням
N або n	Виводить десяткове значення з плаваючою комою, використовуючи при необхідності роздільник - кому між групами по три розряди
X або x	Виводить ціле число в шістнадцятковому вигляді. Шістнадцятиричні цифри вводяться в верхньому або нижньому регістрі в залежності від того, зазначений знак X або x

Це дає достатні знання щодо виведення, щоб продовжити вивчати приклади C++/CLI. Тепер подивимося на застосування описаного в дії.

Нижче наведено демонстраційний приклад виведення консольної програми CLI, що обчислює ціну килимового покриття.

```
int main(array<System::String ^> ^args)
{
	double carpetPriceSqMt = 27.95;
	double roomWidth = 13.5; // В метрах
	double roomLength = 24.75; // В метрах
	const int feetPerMeter = 3;
	double roomWidthMts = roomWidth / feetPerMeter;
```

```
        double roomLengthMts = roomLength / feetPerMeter;
        double carpetPrice = roomWidthMts * roomLengthMts * carpetPriceSqMt;
        Console::WriteLine(L"Розмір кімнати {0:F2} метрів на {1:F2} метрів", roomLengthMts,
roomWidthMts);
        Console::WriteLine(L"Площа кімнати {0:F2} квадратних метрів", roomLengthMts *
roomWidthMts);
        Console::WriteLine(L"Ціна килимового покриття ${0:F2}", carpetPrice);
        return 0;
}
```

Виведення в консолі цієї програми буде наступним

```
Розмір кімнати 8,25 метрів на 4,50 метрів
Площа кімнати 37,13 квадратних метрів
Ціна килимового покриття $1037,64
```

Оператори виведення використовують специфікатор формату F2, щоб обмежити вихідні значення двома десятковими знаками після крапки. Без цього у виведенні може вийти більше знаків, що небажано, особливо коли мова йде про ціну. Щоб побачити різницю, видаліть специфікатор формату.

Оператор виведення площі використовує арифметичний вираз у другому аргументі функції `WriteLine()`. Компілятор спочатку вирахує цей вираз, а потім передасть результат у вигляді аргументу функції. У загальному випадку ви завжди можете використовувати вираз як аргумент функції - до тих пір, поки тип отриманого результату узгоджується з типом параметра функції.

1.3 Клавіатурне введення в C++/CLI

Можливості введення з клавіатури, які є у консольних програм .NET Framework, дещо обмежені. Ви можете прочитати повний рядок введення як текстовий рядок, використовуючи для цього *функцію `Console::ReadLine()`*, або ж прочитати окремий символ, застосовуючи для цього *функцію `Console::Read()`*. Можна також прочитати натискання клавіші за допомогою функції `Console::ReadKey()`.

Функція `Console::ReadLine()` використовується наступним чином.

```
String^ line = Console::ReadLine();
```

Цей оператор читає весь вхідний рядок тексту, завершений натисканням клавіші <Enter>. Змінна `line` має тип `String^` і зберігає посилання на рядок, який виходить в результаті виконання функції `Console::ReadLine()`. Символ ^ після імені типу `String` вказує, що це - *дескриптор* (handle), який посилається на об'єкт типу `String`.

Оператор, який читає один символ з клавіатури, виглядає наступним чином.

```
char ch = Console::Read();
```

За допомогою функції `Read()` можете читати вхідні дані символ за символом, а потім проаналізувати прочитані символи і перетворити їх у відповідні числові значення.

Функція `Console::ReadKey()` повертає натиснуту клавішу у вигляді об'єкта класу `ConsoleKeyInfo`, який представляє собою клас типу значення, визначений у просторі імен `System`. Нижче наведено оператор читання натиснутої клавіші.

```
ConsoleKeyInfo keyPress = Console::ReadKey(true);
```

Аргумент `true` функції `ReadKey()` пригнічує відображення натиснутої клавіші в командному рядку. Аргумент `false` (або відсутність аргументу) змушує функцію відображати символ натиснутої клавіші. Результат виконання функції зберігається в змінній `keyPress`. Щоб ідентифікувати символ, відповідний натиснутій клавіші (або клавішам), необхідно застосовувати вираз `keyPress.KeyChar`. Таким чином, щоб вивести повідомлення, яке показує символ натиснутої клавіші, скористайтеся наступним оператором.

```
Console::WriteLine(L"Натиснута клавіша відповідає символу: {0}", keyPress.KeyChar);
```

Натиснута кнопка ідентифікується виразом `keyPress.Key`. Цей вираз посилається на значення з перерахування C++/CLI, що ідентифікує натиснуту клавішу.

Хоча те, що в консольних програмах C++/CLI немає форматування введення, може здатися незручним під час вивчення, на практиці це обмеження несуттєве. Майже всі реальні програми, які вам доведеться писати, приймають введення через компоненти вікна, тому зазвичай немає необхідності читати дані з командного рядка.

Читання числових значень з командного рядка передбачає використання деяких засобів.

Якщо ви читаєте рядок, що містить цілочисельне значення з використанням функції `Console::ReadLine()`, *функція `Parse()`* в класі `Int32` перетворює його автоматично в 32-бітове ціле. Ось як можна читати цілочисельне значення.

```
Console::Write(L"Введіть ціле число:");
int value = Int32::Parse(Console::ReadLine());
Console::WriteLine(L"Ви ввели {0}", value);
```

Перший оператор просто запрошує ввести необхідне значення, а другий читає введення. Рядок, який повертається функцією `Console::ReadLine()`,

передається у вигляді аргументу функції `Parse()`, що відноситься до класу `Int32`. Це перетворює рядок у 32-бітове ціле і зберігає результат у змінній `value`. Останній оператор виводить значення, щоб показати, що все пройшло правильно. Звичайно, якщо ви введете щось відмінне від цілого числа, виникне помилка.

Інші класи значень, що відповідають базовим «рідним» типам C++, також визначають функцію `Parse()`, тому, наприклад, коли захочете прочитати значення з плаваючою комою з клавіатури, передайте функції `Console::Parse()` рядок, який повертає функція `Console::ReadLine()`. Результатом буде значення типу `double`.

1.4 Об'єкти класу String

Тип класу `String`, визначений у просторі імен `System`, являє рядок в `C++/CLI` (фактично рядки складаються їх символів `Unicode`). Він представляє рядок, що складається з послідовності символів типу `System::Char`. Об'єкти класу `String` мають більші можливості, що значно полегшує обробку рядків. Почнемо з початку - зі створення рядка.

Об'єкт `String` можна створити наступним чином.

```
System::String^ saying(L"Хто в роботі, той і в турботі.");
```

Змінна `saying` - відстежуваний дескриптор, який посилається на об'єкт `String`, який ініціалізований рядком в дужках. Ви завжди повинні використовувати відстежувані дескриптори для збереження посилань на об'єкти `String`. Представлений тут строковий літерал є рядком, що складається з широких символів, оскільки він використовує префікс `L`. Якщо пропустити префікс `L`, то вийде строковий літерал, що складається з 8-бітових символів, а так компілятор гарантує, що він буде перетворений в рядок широких символів.

Ви можете звертатися до індивідуальних символів в рядку, використовуючи синтаксис масивів, причому перший символ рядка має індекс `0`. Ось як можна було вивести третій символ рядка `saying`.

```
Console::WriteLine("Третій символ в рядку: {0}", saying[2]);
```

Ви зможете лише читати окремі символи рядка, звертаючись до них за індексом; оновлювати рядок подібним чином не вдасться. Строкові об'єкти є *незмінними* (`immutable`), а отже, не можуть бути змінені.

Ви можете отримати кількість символів рядка, звернувшись до її властивості `Length`. Вивести довжину рядка `saying` можна за допомогою оператора

```
Console::WriteLine("Рядок містить {0} символів.", saying->Length);
```

Оскільки `saying` - це відстежуваний дескриптор (який, як ви знаєте, є різновидом покажчика), необхідно використовувати оператор `->`, щоб звернутися до властивості `Length` (або будь-якого іншого члена об'єкта). Детальніше про властивості ви дізнаєтеся, коли ми почнемо детальний розгляд класів C++/CLI.

1.5 Об'єднання рядків

Для об'єднання рядків можна використовувати *оператор конкатенації +*, що дозволяє сформувати новий об'єкт `String`. Розглянемо приклад.

```
String^ name1(L"День");
String^ name2(L"нічь");
String^ name3(name1 + L" і " + name2);
```

Після виконання цих операторів змінна `name3` містить рядок `L"День і нічь"`. Зверніть увагу на використання оператора `+` для об'єднання об'єктів `String` із строковими літералами. Також ви можете об'єднувати об'єкти `String` з числовими або логічними значеннями, які при цьому автоматично перетворюються в строковий вид. Наступні оператори ілюструють сказане.

```
String^ str(L"Значення: ");
String^ str1(str + 2.5); // Результат - новий рядок "Значення: 2,5"
String^ str2(str + 25); // Результат - новий рядок " Значення: 25"
String^ str3(str + true); // Результат - новий рядок "Значення: True"
```

Можна також з'єднувати рядки з символом, але результат при цьому залежить від типу символу.

```
char ch('Z');
wchar_t wch(L'Z ');
String^ str4(str + ch); // Результат - новий рядок "Значення: 90"
String^ str5(str + wch); // Результат - новий рядок "Значення: Z"
```

У коментарях відображені результати цих операцій. Символ типу `char` трактується як числове значення, так що ви отримуєте код символу, приєднаний до рядка. Символ `wchar_t` має той же тип, що і символи в об'єкті `String` (тип `Char`), тому символ додається до рядка.

Не забувайте, що об'єкти `String` незмінні; будучи одного разу створені, змінюватися вони не можуть. Це означає, що всі операції, які, ймовірно, змінюють об'єкти `String`, завжди в результаті породжують нові об'єкти `String`.

У класі `String` також визначена функція `Join()`, яку ви можете використовувати, коли хочете об'єднати серію рядків, збережених в масиві, в

один загальний рядок з роздільниками. Нижче показано, як можна об'єднати разом імена, розділивши їх комами.

```
array<String^>^ names = {L"Дарина", L"Микола", L"Хрестина", L"Василина", L"Кирило"};
String^ separator(L", ");
String^ joined = String::Join(separator, names);
```

Після виконання цих операторів змінна `joined` посилається на рядок

```
L"Дарина, Микола, Хрестина, Василина, Кирило"
```

Рядок `separator` вставляється між об'єднуваними фрагментами - вихідними рядками масиву `names`. Зрозуміло, рядок-роздільник може бути будь-який, який вам сподобається (наприклад, він міг би бути `L" і "`, тоді в результаті вийшло б `L"Дарина і Микола і Хрестина і Василина і Кирило"`).

1.6 Зміна рядків

Нерідко виникає необхідність відсікання пробілів на початку і в кінці рядка. Це завдання вирішує *функція* `Trim()` об'єкта `String`.

```
String^ str = {L"    За один раз не зітнеш дерева враз.       "};
String^ newStr = str->Trim();
```

Функція `Trim()` у другому операторі видаляє всі пробіли з початку і кінця рядка `str`, а як результат повертає новий об'єкт `String`, посилання на який міститься в змінній `newStr`. Звичайно, якщо вам не потрібно зберігати початкове значення рядка, то можете помістити результат назад, в змінну `str`.

Існує й інша версія функції `Trim()`, що дозволяє вказати символи, які повинні бути виключені з початку і кінця рядка. Ця функція дуже гнучка, оскільки представляє кілька способів вказування символів, які треба видалити. Ви можете задати їх в масиві і передати дескриптор масиву як аргумент функції.

```
String^ toBeTrimmed(L"wool wool sheep sheep wool wool wool");
array<wchar_t>^ notWanted = { L'w', L'o', L'l', L' ' };
Console::WriteLine(toBeTrimmed->Trim(notWanted));
```

Тут ми маємо рядок `toBeTrimmed`, який складається з `"sheep"` (вівці), покритої `"wool"` (шерстю). Масив символів рядка для видсікання визначено в змінній `notWanted`, тому передача його функції `Trim()` видаляє будь-які символи, що містяться в масиві, з обох боків рядка доки не зустрінеться будь-який інший, не вказаний у функції `Trim()` символ. Нагадаємо, що об'єкти `String` незмінні, тому вихідний рядок залишиться колишнім, а функція `Trim()`

поверне новий екземпляр рядка. Виконання цього фрагмента коду дасть наступне виведення.

```
sheep sheep
```

Якщо ви вкажете символьні літерали без префікса L, вони будуть мати тип char (що відповідає класу типу значення SByte), але компілятор неявно перетворює їх в тип wchar_t.

Ви можете також вказати символи, які функція Trim() повинна видалити, як окремі аргументи, тому останній оператор попереднього фрагмента можна переписати таким чином.

```
Console::WriteLine(toBeTrimmed->Trim(L'w', L'o', L'l', L' '));
```

Результат буде, як і у попередній версії оператора. Ви можете передавати стільки аргументів типу wchar_t, скільки хочете, але якщо символів занадто багато, то масив буде більш вдалим рішенням.

Якщо хочете виконати усічення тільки з одного кінця рядка, використовуйте *функцію TrimEnd()* або *TrimStart()*. Вони мають такі ж версії, як і функція Trim(), тому без аргументів усікають пробіли, з аргументом-масивом - відсікають символи, що містяться в масиві, а з явними аргументами wchar_t видаляють символи, передані в аргументах.

Протилежність усіканню рядка - його доповнення з будь-якого кінця пробілами або іншими символами. Для виконання цієї операції передбачені *функції PadLeft()* і *PadRight()*, які доповнюють рядок ліворуч і праворуч відповідно. Головне призначення цих функцій - форматування виведення, коли потрібно вивести рядки, вирівнюючи їх праворуч або ліворуч в межах фіксованої ширини. Найпростіші версії функцій PadLeft() і PadRight() отримують один аргумент, який вказує довжину рядка, яка повинна вийти в результаті цієї операції.

Розглянемо приклад.

```
String^ value(L"3.142");
String^ leftPadded(value->PadLeft(10));          // Результат: L"     3.142"
String^ rightPadded(value->PadRight(10));        // Результат: L"3.142     "
```

Якщо довжина, передана в аргументі, менше або дорівнює довжині початкового рядка, обидві функції повертають новий об'єкт String, ідентичний оригіналу.

Щоб доповнити рядок символом, відмінним від пробілу, можна вказати потрібний символ в якості другого аргументу функцій PadLeft() і PadRight(). Нижче наведено кілька прикладів.

```
String^ value(L"3.142");
String^ leftPadded(value->PadLeft(10, L'*'));          // Результат: L "*****3.142"
String^ rightPadded(value->PadRight(10, L'#')); // Результат: L "3.142#####"
```

Звичайно, у всіх цих випадках можна помістити результуючий рядок назад в дескриптор, який посилається на вихідний рядок, що призведе до втрати вихідної версії рядка.

Клас `String` володіє також *функціями `ToUpper()`* і *`ToLower()`*, призначеними для перетворення всього рядка в верхній і нижній регістр відповідно. Ось як це працює.

```
String^ proverb(L"Бджола мала, а й та працює.");
String^ upper(proverb->ToUpper());       // Результат: "БДЖОЛА МАЛА, А Й ТА ПРАЦЮЄ."
```

Функція `ToUpper()` повертає новий рядок, який повторює вихідний, але перетворює все його символи в прописні.

Функція `Insert()` служить для вставки рядка в задану позицію існуючого рядка. Ось приклад цієї операції.

```
String^ proverb(L"Хто що знає, тим і заробляє.");
String^ newProverb = proverb->Insert(19, L"хліб ");
```

Функція вставляє рядок, зазначену в другому аргументі, у вихідну рядок, починаючи з позиції, переданої першим аргументом. В результаті виходить новий рядок такого змісту.

```
Хто що знає, тим і хліб заробляє.
```

Можна також замінити всі входження заданого символу або підрядка в рядку іншим символом або підрядком. Наступний фрагмент коду ілюструє цю можливість.

```
String^ proverb(L"Без діла жить — тільки небо коптить.");
Console::WriteLine(proverb->Replace(L' ', L'*'));
Console::WriteLine(proverb->Replace(L"жить — тільки небо коптить.", L"слабіє сила."));
```

Виконання цього фрагмента дасть наступне виведення.

```
Без*діла*жить*—*тільки*небо*коптить.
Без діла слабіє сила.
```

Перший аргумент *функції `Replace()`* задає символ або підрядок, який має бути замінений, а другий аргумент визначає заміну.

1.7 Порівняння рядків

Ви можете порівнювати два об'єкти `String`, використовуючи *функцію `Compare()`* класу `String`. Функція повертає ціле число менше нуля, рівне нулю або більше нуля, в залежності від того, менше перший аргумент другого, дорівнює або більше його.

Ось приклад.

```
String^ him(L"Данило");
String^ her(L"Дарина");
int result(String::Compare(him, her));
if (result <0)
	Console::WriteLine(L"{0} менше ніж {1}.", him, her);
else if (result > 0)
	Console::WriteLine(L"{0} більше ніж {1}.", him, her);
else
	Console::WriteLine(L"{0} еквівалентно {1}.", him, her);
```

Ви зберігаєте ціле число, повернене функцією `Compare()`, у змінній `result` і використовуєте її в операторі `if`, щоб вибрати відповідний висновок. Виконання цього фрагмента дасть наступне виведення.

```
Данило менше ніж Дарина.
```

Існує й інша версія функції `Compare()`, яка вимагає третього аргументу типу `bool`. Якщо в третьому аргументі передається значення `true`, то рядки, на які посилаються перші два аргументи, порівнюються без урахування регістру; якщо ж третій аргумент дорівнює `false`, то поведінка буде така, як і у попередній версії функції `Compare()`.

1.8 Пошук рядків

Можливо, найпростіша операція пошуку - це перевірка того, починається або ж закінчується заданий рядок зазначеним підрядком. Це роблять *функції `StartsWith()`* і *`EndsWith()`*. Обом функціям передається дескриптор підрядка, який потрібно знайти, і вони повертають логічне значення, яке вказує на присутність підрядка в заданому рядку. Ось фрагмент коду, який демонструє застосування функції `StartsWith()`.

```
String^ sentence(L"Життя не має ціни, а воля дорожча за життя.");
if (sentence->StartsWith(L"Життя"))
	Console::WriteLine(L"Фраза починається з 'Життя'.");
```

Виконання цього фрагмента дасть таке виведення.

```
Фраза починається з 'Життя'.
```

Зрозуміло, ви можете також застосувати функцію `EndsWith()` до рядка `sentence`.

```
Console::WriteLine("Фраза {0} закінчується словом 'життя'.",
        sentence -> EndsWith(L"життя") ? L"" : L"не");
```

Результат виразу з умовним оператором вставляється у вихідний рядок. Це буде порожній рядок, якщо функція `EndsWith()` поверне значення `true`, і `"не"`, якщо вона поверне значення `false`. В даному випадку функція повертає значення `false` (через точку в кінці початкового рядка).

Функція `IndexOf()` шукає в рядку перше входження зазначеного символу або підрядка і повертає індекс входження, якщо воно є, або `-1` інакше. Шуканий символ або підрядок передається в аргументі функції. Розглянемо приклад.

```
String^ sentence(L"Козача потилиця панам не хилиться.");
int yaPosition(sentence->IndexOf(L'я'));        // Повертає 14
int nePosition(sentence->IndexOf(L"не"));       // Повертає 22
```

У першому випадку здійснюється пошук літери `"я"`, а в другому - слова `"не"`. Функцією `IndexOf()` повертаються значення вказані в коментарях.

Набагато частіше вам знадобиться знаходити всі входження заданого символу або підрядка. Для цього підійде інша версія функції `IndexOf()`, яка призначена для повторюваного виклику. Вона отримує другий аргумент, який вказує індекс позиції початку пошуку. Нижче наведено приклад застосування цієї версії функції.

```
String^ words(L"Козак не боїться ні тучі, ні грому, ні хмари, ні чвари.");
String^ word(L"ні");
int index(0);
int count(0);
while ((index = words->IndexOf(word, index)) >= 0)
{
        index += word->Length;
        ++count;
}
Console::WriteLine(L"Слово '{0}' знайдено {1} рази в рядку: \n{2}", word, count, words);
```

Цей фрагмент підраховує кількість входжень фрагмента `"ні"` в рядку `words`. Операція пошуку виконується в умові циклу `while`, а результат зберігається в змінній `index`. Цикл повторюється до тих пір, поки значення `index` додатнє, тому, коли функція `IndexOf()` повертає значення `-1`, цикл завершується. У тілі циклу значення `index` збільшується на довжину слова `word`, що переміщує позицію початку пошуку на символ, наступний за знайденим екземпляром підрядка `word`, готуючи наступну ітерацію пошуку. Значення змінної `count` збільшується в циклі, тому, коли цикл завершується, вона містить

накопичену загальну кількість входжень рядка word в рядку words. В результаті виконання цього фрагмента виходить таке виведення.

```
Слово 'ні' знайдено 4 рази в рядку:
Козак не боїться ні тучі, ні грому, ні хмари, ні чвари.
```

Функція LastIndexOf() у всьому подібна до функції IndexOf, за винятком того, що виконує пошук в зворотному порядку, починаючи з кінця рядка або із заданою позиції. Ось як операція, яка виконується попереднім фрагментом, може бути реалізована за допомогою функції LastIndexOf().

```
int index(words->Length - 1);
int count(0);
while (index >= 0 && (index = words->LastIndexOf(word, index)) >= 0)
{
        --index;
        ++count;
}
```

За умови, що значення рядків word і words ті ж самі, що і раніше, цей фрагмент створює той же результат. Оскільки функція LastIndexOf() шукає в зворотному порядку, тут початковою позицією пошуку є останній символ рядка, який має індекс words -> Length - 1. Коли знайдено входження рядка word, значення змінної index зменшується на 1, так що наступний зворотний пошук починається з символу , що передує поточному входженню рядка word. Якщо рядок word зустрінеться прямо на початку рядка words (в позиції індексу 0), то зменшення значення змінної index призведе до того, що він стане рівним -1, що не є допустимим аргументом для функції LastIndexOf(), оскільки початкова позиція пошуку завжди повинна знаходитися в рядку. Додаткова перевірка від'ємного значення змінної index в умові циклу запобігає описаній ситуації. Якщо лівий операнд оператора && дорівнює false, то правий операнд не перевіряється.

І остання функція пошуку, про яку слід згадати, - це *функція IndexOfAny()*; вона шукає в рядку перше входження будь-якого символу з масиву array<wchar_t>, який передається в аргументі. Подібно функції IndexOf(), ця функція має версії, які шукають від початку рядка або від зазначеної позиції індексу.

1.9 Масиви середовища CLR

Масиви середовища CLR відрізняються від масивів «рідного» C++. Пам'ять для масиву CLR виділяється в очищуваній динамічній пам'яті, але це ще не все. Як ви незабаром побачите, масиви CLR володіють вбудованими можливостями, яких немає у масивів «рідного» C++. Тип змінної масиву вказано ключовим

словом array. При цьому ви також повинні вказати тип елементів масиву в кутових дужках, наступних за ключовим словом array, тому загальна форма змінної, що посилається на одновимірний масив, - array<тип_елементу>^. Оскільки масиви CLR створюються в динамічній пам'яті, змінні масивів - це завжди відстежувані дескриптори. Ось приклад оголошення змінної масиву.

```
array<int>^ data;
```

Змінна масиву data може зберігати посилання на одновимірний масив елементів типу int.

Ви можете створити масив CLR, використовуючи оператор gcnew, одночасно з оголошенням змінної масиву.

```
array<int>^ data = gcnew array<int>(100); // Створити масив для зберігання 100 цілих чисел
```

Цей оператор створює одновимірний масив data (зверніть увагу: змінна масиву - відстежуваний дескриптор, тому не можна забувати про символ ^, який слідує за вказуванням типу елемента, укладеного в кутові дужки). Кількість елементів вказується в круглих дужках, наступних за специфікацією типу масиву, також укладеною в кутові дужки, тобто в даному випадку це буде масив, що містить 100 елементів типу int.

Безумовно, для ініціалізації змінної data можна також використовувати функціональну форму запису.

```
array<int>^ data(gcnew array<int>(100)); // Створити масив для зберігання 100 цілих чисел
```

Як і в масивах «рідного» C++, елементи масивів CLR індексуються починаючи з нуля, тому можете встановити значення елементів масиву data наступним чином.

```
for (int i = 0; i <100; i++)
	data[i] = 2 * (i + 1);
```

Цикл встановлює для елементів значення 2, 4, 6 і так далі до 200. Елементи масиву CLR є об'єктами, тому тут ви зберігаєте в масиві об'єкти типу Int32. Вони поводяться подібно звичайним цілим числам в арифметичних виразах, тому той факт, що вони є об'єктами, в таких ситуаціях не важливий.

У попередньому циклі кількість оброблюваних елементів задано літеральним значенням. Але краще використовувати властивість Length масиву, яка зберігає кількість його елементів.

```
for (int i = 0; i < data->Length; i++)
	data[i] = 2 * (i + 1);
```

Для звернення до властивості `Length` тут використовується оператор `->`, тому що `data` - відстежуваний дескриптор, який працює подібно покажчику. Ця властивість зберігає кількість значень як 32-бітове ціле. У міру необхідності ви можете отримати довжину масиву як 64-бітове ціле за допомогою властивості `LongLength`.

Можна також перебрати всі елементи масиву, скориставшись *циклом `for each`* середовища CLR. Даний цикл є певним аналогом *серійного оператора `for`* «рідного» C++, який призначений для використання з масивами CLR.

```
array<int>^ values = { 3, 5, 6, 8, 6 };
for each (int item in values)
{
	item = 2 * item + 1;
	Console::Write("{0,5}", item);
}
```

Перший оператор демонструє можливість ініціалізації дескриптора масиву за допомогою масиву, визначеного набором значень. Розмір масиву визначається кількістю вихідних значень в фігурних дужках (в даному випадку 5), а значення присвоюються елементам послідовно. Таким чином, дескриптор `values` посилатиметься на масив з 5 цілих чисел, елементи якого містять значення 3, 5, 6, 8 і 6. У циклі змінна `item` посилається на кожен елемент масиву по черзі. Перший оператор в тілі циклу замінює значення поточного елемента його подвоєним старим значенням плюс 1. Другий оператор циклу виводить нове значення, вирівнюючи його по правій стороні поля, шириною 5 символів, тому в результаті виходить наступне виведення.

```
    7   11   13   17   13
```

Те, що відбувається тут не очевидно. Цикл `for each` не змінює елементи в масиві `value`. Змінна `item` звертається до значення кожного елемента масиву по черзі, але вона не посилається на елементи масиву безпосередньо.

Змінна масиву може зберігати адресу будь-якого масиву того ж рангу (ранг - це кількість вимірювань, який в разі масиву `data` дорівнює 1) і типу елементів. Розглянемо приклад.

```
data = gcnew array<int>(45);
```

Цей оператор створює новий одновимірний масив з 45 елементів типу `int` і зберігає його адресу в дескрипторі `data`. Вихідний масив, на який посилався дескриптор `data`, відкидається.

Звичайно, елементи масиву можуть бути будь-якого типу, тому дуже легко можна створити масив рядків.

```
array<String^>^ names = { "Jack", "Jane", "Joe", "Jessica", "Jim", "Jamala" };
```

Елементи цього масиву ініціалізуються рядками, які укладені у фігурні дужки, і кількість цих рядків визначає число елементів в масиві. Об'єкти `String` розміщуються в динамічній пам'яті середовища CLR, тому типом елементів цього масиву є тип відстежуваних дескрипторів - `String^`.

Якщо ви оголосите змінну масиву, не ініціалізувавши її, то повинні потім явно створити масив з переліком ініціалізуючих значень. Розглянемо приклад.

```
array<String^>^ names; // Оголошення змінної масиву
names = gcnew array<String^>{"Jack", "Jane", "Joe", "Jessica", "Jim", "Jamala"};
```

Перший оператор створює змінну `names` типу `Array`, яка за замовчуванням буде ініціалізована значенням `nullptr`. Другий оператор створює масив і ініціалізує його рядками в фігурних дужках. Без явного виклику оператора `gcnew` цей код не компілюється.

Для обнулення будь-якої послідовності числових елементів масиву можете використовувати функцію `Clear()`, яка визначена в класі `Array`. При виклику статичної функції вказують ім'я класу. Нижче показаний приклад виклику функції `Clear()`.

```
Array::Clear(samples, 0, samples->Length); // Обнулити всі елементи
```

Перший аргумент функції `Clear()` - це масив, який повинен бути обнулений; другий - індекс першого елемента, з якого починається обнулення; третій - кількість елементів, що підлягають очистці. Таким чином, в цьому прикладі всім елементам масиву `samples` присвоюється значення `0.0`. Якщо ви застосуєте функцію `Clear()` до масиву дескрипторів, таких як `String^`, то елементи будуть встановлені в значення `nullptr`, а якщо застосувати її до масиву об'єктів типу `bool`, то вони отримають значення `false`.

Тепер саме час розглянути приклад використання масиву CLR.

```
// Проект_КПР1_1.cpp: главный файл проекта.

#include "stdafx.h"
using namespace System;

int main(array<System::String ^> ^args)
{
	array<double>^ samples = gcnew array<double>(50);
	// Створити випадкові значення елементів
	Random^ generator = gcnew Random;
	for (int i = 0; i < samples->Length; i++)
		samples[i] = 100.0*generator->NextDouble();
	// Вивести вміст samples
	Console::WriteLine(L"Масив містить наступні значення:");
	for (int i = 0; i < samples->Length; i++)
	{
```

```
                Console::Write(L"{0,10:F2}", samples[i]);
                if ((i + 1) % 5 == 0)
                        Console::WriteLine();
        }
        // Знайти максимальне значення
        double max(0);
        for each(double sample in samples)
                if (max < sample)
                        max = sample;
        Console::WriteLine(L"Максимальне значення в массиві дорівнює {0:F2}", max);
        return 0;
}
```

Типове виведення даної програми виглядатиме наступним чином:

```
Масив містить наступні значення:
     94,87     37,68     12,86     44,02     95,27
     27,29     24,43     70,93     87,90     57,73
     11,26     90,93     71,48     19,60     78,37
     85,04     34,81     17,36     48,42     70,81
     68,88     45,75     15,24     24,67     95,56
     96,99     40,96     71,64     76,24     36,98
     77,48     25,91     11,67     43,61     56,56
      8,04     20,27     44,23     74,99     15,14
      9,33     92,48     56,99      3,22     43,81
     72,51     92,15     27,52     12,40     26,37
Максимальне значення в массиві дорівнює 96,99
```

Спочатку створюється масив, здатний зберігати 50 елементів типу double.

```
array<double>^ samples = gcnew array<double>(50);
```

Змінна samples типу array повинна бути відстежуваним дескриптором, бо масиви CLR створюються в динамічній пам'яті, контрольованій збирачем «сміття».

Наступні оператори заповнюють масив псевдовипадковими значеннями типу double.

```
Random^ generator = gcnew Random;
for (int i = 0; i < samples->Length; i++)
        samples[i] = 100.0*generator->NextDouble();
```

Перший оператор створює в динамічної пам'яті середовища CLR об'єкт типу Random. Об'єкт Random має функції створення псевдовипадкових значень. Тут в циклі використовується функція NextDouble(), яка повертає випадкове значення типу double, що лежить в межах від 0,0 до 1,0. Помноживши це значення на 100,0, отримаємо значення 0,0 - 100,0. Цикл for зберігає випадкове значення в кожному елементі масиву samples.

Об'єкт Random має також функцію Next(), яка повертає випадкове позитивне значення типу int. Якщо передати цілочисельний аргумент при виконанні функції Next(), то вона поверне випадкове позитивне значення,

менше значення цього аргументу. Можна також передати два цілочисельних аргументи, що представляють мінімальне і максимальне значення, в межах яких має знаходитися випадкове число.

Наступний цикл виводить вміст масиву по п'ять елементів в рядку.

```
Console::WriteLine(L"Масив містить наступні значення:");
for (int i = 0; i < samples->Length; i++)
{
	Console::Write(L"{0,10:F2}", samples[i]);
	if ((i + 1) % 5 == 0)
		Console::WriteLine();
}
```

У циклі значення кожного елемента виводиться в поле шириною 10 символів з двома десятковими розрядами. Покажчик ширини поля гарантує вирівнювання значень в стовпцях. Символ нового рядка виводиться щоразу, коли вираз (i + 1)%5 повертає значення 0, що відбувається після виведення значення кожного п'ятого елемента, тому виходить по п'ять значень в рядку.

І нарешті, максимальне значення елементів масиву визначається наступним чином.

```
double max(0);
for each(double sample in samples)
	if (max < sample)
		max = sample;
```

Тут цикл for each застосовується тільки для демонстрації його можливостей. Цикл порівнює значення змінної max з кожним елементом по черзі, і коли виявляється, що елемент більше поточного значення змінної max, то його значення присвоюється змінній max, і в кінцевому підсумку змінна max містить максимальне з усіх значень елементів масиву.

Можете використовувати тут цикл for, якщо хочете записати позицію індексу максимального елемента разом з його значенням.

```
double max = 0;
int index = 0;
for (int i = 0; i < samples->Length; i++)
	if (max < samples[i])
	{
		max = samples[i];
		index = i;
	}
```

1.10 Сортування одномірних масивів

У класі Array з простору імен System визначена функція Sort(), яка сортує елементи одновимірного масиву так, що вони розташовуються в порядку

зростання. Щоб впорядкувати масив, досить передати його дескриптор функції `Sort()`. Приклад наведено нижче.

```
array <int>^ samples = { 27, 3, 54, 11, 18, 2, 16 };
Array::Sort(samples);              // Сортувати елементи масиву
for each (int value in samples)    // Вивести елементи масиву
      Console::Write(L"{0, 8}", value);
Console::WriteLine();
```

Виклик функції `Sort()` перевпорядковує значення елементів масиву `samples` за зростанням. Результат виконання цього фрагмента буде таким.

```
       2       3       11      16      18      27      54
```

Можна також сортувати діапазон елементів масиву, передавши в двох аргументах функції `Sort()` індекс першого елемента і кількість елементів, що підлягають сортуванню. Розглянемо приклад.

```
array <int>^ samples = { 27, 3, 54, 11, 18, 2, 16 };
Array::Sort(samples, 2, 3); // Сортувати елементи з 2 по 4
```

Цей код сортує три елементи масиву `samples`, починаючи з позиції 2. Після виконання наведених вище операторів елементи масиву будуть мати наступні значення.

```
      27       3       11      18      54      2       16
```

Існує також кілька інших версій функції `Sort()`, які описані в документації, розглянемо одну з них, яка особливо корисна. Ця версія припускає, що у вас є два масиви, які асоційовані так, що елементи першого з них представляють собою ключі до відповідних елементів другого масиву. Наприклад, імена людей можна зберегти в одному масиві, а вагу кожного - у другому. Функція `Sort()` сортує масив `names` в порядку зростання і також змінює порядок елементів масиву `weights`, так що показники ваги як і раніше відносяться до відповідних персон. Розглянемо це на прикладі.

Наступний приклад створює масив імен, зберігає вагу кожної людини у відповідному елементі другого масиву, а потім сортує обидва масиви в одній операції.

```
// Проект_КПР1_2.cpp: главный файл проекта.
// Сортування масивів за ключем (імена) та пов'язаного з ним масиву вага
#include "stdafx.h"

using namespace System;

int main(array<System::String^> ^args)
{
```

```
    array<String^>^ names ={"Захар", "Орися", "Остап", "Росава", "Назар", "Одарка", "Нестор"};
    array<int>^ weights = { 87, 58, 83, 66, 74, 61, 78 };
    Array::Sort(names, weights);                    // Сортувати масиви
    for each(String^ name in names)                 // Вивести імена
        Console::Write(L"{0, 10}", name);
    Console::WriteLine();
    for each(int weight in weights)                 // Вивести вагу
        Console::Write(L"{0, 10}", weight);
    Console::WriteLine();
    return 0;
}
```

Виведення даної програми буде наступним.

```
    Захар      Назар     Нестор     Одарка      Орися      Остап     Росава
       87         74         78         61         58         83         66
```

Значення масиву `weights` відповідають вазі персон в тих же позиціях індексу, що і їх імена в масиві `names`. Викликана тут функція `Sort()` сортує обидва масиви, використовуючи перший з них (`names`) як ключ сортування, що визначає порядок обох масивів. З результату роботи програми видно, що після сортування імені кожної персони відповідає її вага з другого масиву.

1.11 Пошук в одномірному масиві

У класі `Array` передбачені також функції пошуку елементів в одновимірному масиві. Версії функції `BinarySearch()` використовують алгоритм бінарного пошуку для знаходження позиції індексу заданого елемента в усьому масиві або в зазначеному діапазоні його елементів. Алгоритм бінарного пошуку вимагає, щоб елементи були впорядковані, тому перед виконанням пошуку в масиві їх слід впорядкувати.

Пошук в усьому масиві можна виконувати так.

```
array<int>^ values = { 23, 45, 68, 94, 123, 127, 150, 203, 299 };
int toBeFound = 94;
int position = Array::BinarySearch(values, toBeFound);
if (position < 0)
        Console::WriteLine(L"{0}, не знайдено.", toBeFound);
else
        Console::WriteLine(L"{0} знайдено в позиції індексу {1}.", toBeFound, position);
```

Шукане значення зберігається в змінній `toBeFound`. Перший аргумент функції `BinarySearch()` - це дескриптор масиву, в якому виконується пошук, а другий вказує те, що необхідно знайти. Результат пошуку повертається функцією `BinarySearch()` як значення типу `int`. Якщо другий аргумент функції знайдений в масиві, зазначеному в її першому аргументі, повертається індекс його позиції, в іншому випадку повертається від'ємне значення. Таким чином, ви можете перевірити значення повернення, щоб визначити, чи знайдено потрібне

значення в масиві. Оскільки значення в масиві `values` вже впорядковані за зростанням, немає необхідності сортувати його перед початком пошуку. Наведений фрагмент коду створює наступне виведення.

```
94 знайдено в позиції індексу 3.
```

Щоб шукати в заданому діапазоні елементів масиву, можна застосувати версію функції `BinarySearch()` з чотирма аргументами. Перший аргумент - це дескриптор масиву, в якому виконується пошук; другий - позиція індексу елемента, з якого потрібно починати пошук; третій - кількість елементів, серед яких потрібно шукати; четвертий - власне шукане значення. Нижче показано, як це можна використовувати.

```
array<int>^ values = { 23, 45, 68, 94, 123, 127, 150, 203, 299 };
int toBeFound = 127;
int position = Array::BinarySearch(values, 3, 6, toBeFound);
```

Тут здійснюється пошук в масиві, починаючи з його четвертого елемента і до останнього. Результатом пошуку буде наступне:

```
127 знайдено в позиції індексу 5.
```

Як і попередня версія функції `BinarySearch()`, повертає позицію індексу знайденого елемента або негативне значення, якщо пошук не вдався.

Розглянемо приклад.

```
// Проект_КПР1_3.cpp: главный файл проекта.
// Пошук в масивах

#include "stdafx.h"

using namespace System;

int main(array<System::String ^> ^args)
{
	array<String^>^ names = { "Захар", "Орися", "Остап", "Росава", "Назар", "Одарка",
"Нестор", "Богуслава", "Данило", "Тарас" };
	array<int>^ weights = { 87, 58, 83, 66, 74, 61, 78, 57, 85, 72 };
	array<String^>^ toBeFound = { "Орися", "Назар", "Богуслава", "Кирило" };
	Array::Sort(names, weights);                    // Відсортувати масиви
	int result(0);                                  // Зберігає результуюче значення
	for each(String^ name in toBeFound)             // Пошук ваги за іменем
	{
		result = Array::BinarySearch(names, name);  // Пошук імені в масиві імен
		if (result<0)                               // Перевірка результату пошуку
			Console::WriteLine(L"{0} не знайдений.", name);
		else
			Console::WriteLine(L"{0} має вагу {1} кг.", name, weights[result]);
	}
	return 0;
}
```

Результат виконання цієї програми наведений нижче

```
Орися має вагу 58 кг.
Назар має вагу 74 кг.
Богуслава має вагу 57 кг.
Кирило не знайдений.
```

Створюються два пов'язаних масиви (масив імен та масив відповідних ваг в кілограмах). Крім того, створюється масив `toBeFound`, що містить імена людей, вагу яких потрібно дізнатися.

Масиви `names` і `weights` упорядковано з використанням масиву `names` як ключа сортування. Потім в циклі `for each` виконується пошук в масиві `names` кожного з імен, що містяться в масиві `toBeFound`. Змінна циклу `name` по черзі отримує значення кожного з імен масиву `toBeFound`. У циклі здійснюється пошук поточного імені за допомогою оператора

```
result = Array::BinarySearch(names, name);  // Пошук імені в масиві імен
```

В результаті повертається індекс елемента з масиву `names`, який дорівнює `name`, або негативне значення, якщо такого не знайдено. Результат пошуку перевіряється і виводиться відповідне повідомлення.

```
if (result<0) // Перевірка результату пошуку
	Console::WriteLine(L"{0} не знайдений.", name);
else
	Console::WriteLine(L"{0} має вагу {1} кг.", name, weights[result]);
```

Оскільки порядок елементів масиву `weights` визначається порядком елементів `names`, для отримання результату з масиву `weights` можна звернутися до елементу цього масиву за індексом, знайденим при пошуку змінної `name` в масиві `names`. З виведення програми видно, що значення "Кирило" не було знайдено в масиві `names`.

Коли бінарний пошук завершується невдало, повертається не просто якесь випадкове негативне значення. Насправді це значення являє собою бітове доповнення позиції індексу першого елемента, який більше шуканого значення. Знаючи це, скористайтеся функцією `BinarySearch()`, щоб знайти місце в масиві, куди слід було б вставити новий об'єкт, не порушивши при цьому порядок сортування елементів. Припустимо, ви хочете вставити значення "Кирило" в масив `names`. Позицію індексу, куди він повинен бути вставлений, можна знайти за допомогою наступних операторів.

```
array<String^>^ names = { "Захар", "Орися", "Остап", "Росава", "Назар", "Одарка", "Нестор",
"Богуслава", "Данило", "Тарас" };
Array::Sort(names); // Сортувати масив
String^ name = L"Кирило";
int position = Array::BinarySearch(names, name);
```

```
if (position <0) // Якщо результат негативний,
	position = ~position; // Перекинути біти, щоб отримати індекс вставки
```

Якщо результат пошуку негативний, заміна всіх бітів на протилежне значення (0 на 1 і навпаки) дає позицію індексу, куди слід вставити нове ім'я. Якщо результат позитивний, значить, нове ім'я ідентично імені в позиції з індексом, рівним результату, тому це значення можна використовувати безпосередньо.

Тепер можна скопіювати масив names в новий масив, розмір якого на один елемент більше, і застосувати обчислену позицію для вставки імені у відповідне місце.

```
array<String^>^ newNames = gcnew array <String^>(names->Length + 1);
// Копіювати елементи з names в newNames
for (int i = 0; i <position; i++)
	newNames[i] = names[i];
newNames[position] = name; // Копіювати новий елемент
if (position <names->Length) // Якщо ще залишилися елементи names
	for (int i = position; i < names->Length; i++)
		newNames[i + 1] = names[i]; // Копіювати їх в newNames
```

Це створює новий масив довжиною на один елемент більше, ніж старий. Потім копіюються елементи зі старого масиву в новий масив - від початку до позиції індексу position - 1. Після цього копіюється нове ім'я, а за ним - решта елементів зі старого масиву. Щоб відкинути старий масив, можна написати просто.

```
names = nullptr;
```

Для виведення значень створеного масиву newNames використаємо цикл for each.

```
for each(String^ name in newNames)
	Console::Write(L"{0,10}", name);
```

Результатом буде наступне виведення.

```
 Богуслава     Данило      Захар     Кирило      Назар     Нестор     Одарка      Орися      Остап
Росава      Тарас
```

Повний текст приклада наведений нижче.

```
// Проект_КПР1_4.cpp: главный файл проекта.
// Вставка значення у відсортований масив

#include "stdafx.h"

using namespace System;
```

```
int main(array<System::String ^> ^args)
{
	array<String^>^ names = { "Захар", "Орися", "Остап", "Росава", "Назар", "Одарка",
"Нестор", "Богуслава", "Данило", "Тарас" };
	Array::Sort(names); // Сортувати масив
	String^ name = L"Кирило";
	int position = Array::BinarySearch(names, name);
	if (position <0) // Якщо результат негативний,
		position = ~position; // Перекинути біти, щоб отримати індекс вставки

	array<String^>^ newNames = gcnew array <String^>(names->Length + 1);
	// Копіювати елементи з names в newNames
	for (int i = 0; i <position; i++)
		newNames[i] = names[i];
	newNames[position] = name; // Копіювати новий елемент
	if (position < names->Length) // Якщо ще залишилися елементи names
		for (int i = position; i < names->Length; i++)
			newNames[i + 1] = names[i]; // Копіювати їх в newNames
	names = nullptr;
	for each(String^ name in newNames)
		Console::Write(L"{0,10}", name);
	return 0;
}
```

1.12 Багатомірні масиви

Ви можете створювати масиви з двома і більше вимірами; максимальна кількість вимірів масиву - 32, чого цілком достатньо в більшості випадків. Кількість вимірів масиву вказується в кутових дужках відразу після типу елемента і відділяється від нього комою. За замовчуванням масив має один вимір, ось чому ми не вказували його до сих пір. Як можна створити двовимірний масив цілочисельних елементів, показано нижче.

```
array<int, 2>^ values = gcnew array<int, 2>(4, 5);
```

Цей оператор створює двовимірний масив з чотирьох рядків і п'яти стовпців, так що всього він вміщує 20 елементів. Щоб звернутися до елементу багатовимірного масиву, ви задаєте набір значень індексу - по одному для кожного виміру; вони вказуються в квадратних дужках слідом за ім'ям масиву і розділяються комами. Ось як можна встановити значення елементів двовимірного масиву цілих чисел.

```
int nrows(4);
int ncols(5);
array<int, 2>^ values(gcnew array<int, 2>(nrows, ncols));
for (int i = 0; i < nrows; i++)
	for (int j = 0; j < ncols; j++)
		values[i, j] = (i + 1) * (j + 1);
```

Вкладений цикл перебирає всі елементи масиву. Зовнішній цикл перебирає рядки, а внутрішній - кожен елемент в поточному рядку. Як бачите, значення кожного елемента встановлюється рівним значенню, отриманому в

результаті обчислення виразу `(i + 1) * (j + 1)`. В результаті елементи першого рядка будуть мати значення 1, 2, 3, 4, 5, елементи другого рядка - 2, 4, 6, 8, 10 і т.д., аж до останнього рядка, елементи якої будуть містити значення 4, 6, 12, 16, 20.

Ви напевно помітили, що форма запису доступу до елементів двовимірного масиву відрізняється від використовуваної з масивами «рідного» C++. Це не випадково: масив C++/CLI не є масивом, подібним масиву «рідного» C++. Як уже згадувалося, розмірність масиву називається його *рангом* (rank), тому ранг масиву `values` з попереднього прикладу дорівнює 2. Звичайно, ви можете оголошувати масиви C++/CLI з рангом 3 і більше, аж до масивів з рангом 32. На відміну від цього масиви «рідного» C++ в дійсності завжди мають ранг 1, оскільки «рідні» масиви C++ з двома і більше вимірами насправді є масивами масивів. Як ви побачите пізніше, масиви масивів також можна оголошувати і в C++/CLI.

1.13 Зубчасті масиви

Розглянемо використання зубчастих масивів на прикладі. Необхідно створити консольний додаток, який створює зубчастий масив (масив масивів) з речень, які користувач увів з клавіатури.

Розмірність зовнішнього масиву, який міститиме масиви типу `String` визначатимемо через запит до користувача. Кількість внутрішніх масивів має відповідати кількості введених користувачем речень. Кожне окреме речення, введення якого з клавіатури завершене натисканням клавіши Enter, має сформувати окремий внутрішній масив. Розмірності внутрішніх масивів мають визначатися з кількості слів кожного введеного речення. Кожне окреме слово введеного речення має зберігатися як елемент певного внутрішнього масиву. Значення елементів масиву виводитимемо в консольному вікні у форматованому та чітко структурованому вигляді.

Щоб створити масив масивів потрібно для початку знати розмірність зовнішнього масиву. Для цього робимо запит до користувача:

```
Console::WriteLine(L"Введіть розмірність зовнішнього масиву");
size_t n = Int32::Parse(Console::ReadLine());
```

Розмірність масиву зберігаємо у змінну n типу `size_t`, але дані, що передаються функцією `Console::ReadLine()` мають рядковий тип. Права і ліва частина виразу повинні мати однаковий тип, або права частина виразу має бута такого типу, який допускає автоматичне перетворення в тип лівої частини виразу. В нашому випадку неможливо перетворити автоматично рядковий тип у

беззнаковий цілочисельний. Тому використовуєму функцію перетворення типів Parse з класу цілочисельних значень Int32.

Після цього можемо оголосити і визначити масив:

```
array<array<String^>^>^ масивРеченнь = gcnew array<array<String^>^>(n);
```

Ми оголосили масив, елементами якого будуть масиви, що міститимуть рядкові елементи типу String^. Це і є масив масивів або зубчастий масив, тому що елементами масиву є інші масиви різної довжини.

Після цього організовуємо роботу програми у циклі for, кількість повторень якого відповідає кількості елементів зовнішнього масиву

```
for (size_t i = 0; i < n; i++)
```

В середині циклу оголошуємо змінну, куди будемо запам'ятовувати речення, яке ввів користувач на кожній ітерації циклу.

```
//Зчитуємо введене речення
String^ речення = Console::ReadLine();
```

Слова у масив будемо запам'ятовувати без знаків пунктуації. Кількість слів у реченні будемо визначати за кількістю пробілів між словами. Тому потрібно провести ряд операцій, щоб прибрати зайві пробіли та символи.

Слова у масив будемо запам'ятовувати лише малими літерами. Для цього перетворимо всі символи речення на малі функцією ToLower():

```
//Змінюємо всі літери речення на маленькі
речення = речення->ToLower();
```

Тепер нам потрібно видалити знаки пунктуації. Якщо користувач пропустив пробіл після знака пунктуації, тоді, якщо видалити знак пунктуації, два слова об'єднаються в одне. Щоб уникнути цієї помилки ми будемо не видаляти знаки пунктуації, а замінювати їх на пробіли. Перебір символів рядкового типу String^ можна організувати у циклі, бо змінна типу String^ являє собою масив символів типу char або wchar_t і до них можна звуртатися за допомогою індексу – порядкового номеру символу у ряду.

Але є простіший спосіб з використанням *регулярних виразів (regular expressions)* для текстових даних. Щоб скористатися функціями для регулярних виразів, на початку програми потрібно підключити область імен регулярних виразів:

```
using namespace System::Text::RegularExpressions;
```

Тепер можемо за допомогою однієї функції, без використання циклічного перебору усіх символів речення в програмі, замінити всі символи з заданого набору символів на вказаний підрядок тексту. Для цього використовується функція `Replace()` класу `Regex` з регулярних виразів. В нашому випадку будемо замінювати кожен символ з вказаного набору на пробіл:

```
//Замінюємо в реченні знаки пунктуації на пробіл за допомогою бібліотеки регулярних виразів
речення = Regex::Replace(речення, "[-.?!)(,:;]", " ");
```

Після заміни знаків пунктуації на пробіли, якщо після знаку стояв вже пробіл, то тепер їх буде по два між словами. А якщо між словами користувач вже до того помилково увів два пробіли, то після заміни знаку пунктуації пробілів може стати три. Та як кількість слів ми можемо визначити саме за пробілами між ними, то нам потрібно, щоб між кожними двома словами був один і лише один пробіл. У зв'язку з цим виникає задача видалення зайвих пробілів між словами. Виконати це завдання ми можемо за допомогою функції Replace() класу `String^`, яка замінює один підрядок в тексті на інший.

```
//Замінюємо в реченні потрійні пробіли на одинарні, якщо такі є
речення = речення->Replace("   ", " ");
//Замінюємо в реченні подвійні пробіли на одинарні, якщо такі є
речення = речення->Replace("  ", " ");
```

Видалимо зайві пробіли на початку та в кінці речення, якщо користувач помилково їх увів, та пробіл, яким ми замінили останній знак пунктуації у реченні. Для цього скористаємося функцією `Trim()` класу `String`:

```
//Відкидаємо пробіли на початку та в кінці речення
речення = речення->Trim();
```

Тепер безпосередньо потрібно підрахувати кількість пробілів у реченні, за якими і визначимо кількість слів. Для цього знову використаємо бібліотеку регулярних виразів.

```
//Рахуємо кількість пробілів, що залишились
size_t count(0);
//Створюємо змінну типу Regex і вказуємо рядок, який будемо шукати (пробіл)
Regex^ пробіл = gcnew Regex(" ");
//Створюємо змінну типу MatchCollection і присвоюємо їй результати пошуку співпадінь
MatchCollection^ співпадіння = пробіл->Matches(речення);
//Запам'ятовуємо в змінну count кількість знайдених пробілів в реченні + 1
count = співпадіння->Count + 1;
```

Тут ми створили:

- змінну `count` типу `size_t`, в яку будемо запам'ятовувати знайдену кількість пробілів;
- змінну `пробіл` типу `Regex^`, якій присвоїли підрядок, що потрібно знайти (в нашому випадку це пробіл);
- змінну `співпадіння` типу `MatchCollection^`, якій присвоюємо результат пошуку співпадінь вказаного підрядку у заданому тексті за допомогою функції `Matches()` класу `Regex^`.

Кількість співпадінь дізнаємося з властивості `Count` змінної `співпадіння` класу `MatchCollection^`.

Тепер можемо створити внутрішній масив для збереження слів поточного речення. Розмірність цього масиву дорівнює кількості знайдених пробілів плюс 1, тому що після останнього слова в реченні пробілу немає.

```
//Створюємо масив розмірність якого дорівнює кількості слів у реченні
масивРеченнь[i] = gcnew array<String^>(count);
```

В подальшому будемо перебирати у циклі всі символи речення окрім пробілів і формувати їх у слова для запису у внутрішній масив.

```
String^ слово = "", ^ останнєСлово = "";
for each (wchar_t літера in речення)
{
	if (літера != ' ')
	{
		слово += літера;
		останнєСлово = слово;
	}
	else
	{
		масивРеченнь[i][index] = слово;
		index++;
		слово = "";
	}
}
//Оскільки після останнього слова у реченні немає пробілу, записуємо його окремо після
роботи циклу
масивРеченнь[i][index] = останнєСлово;
```

До кожного символу речення звертаємося у циклі `for each` через ідентифікатор `літера` типу `wchar_t`. Якщо цей символ не пробіл, додаємо його до вже існуючих символів у змінній `слово` типу `String^`. Якщо символ виявляється пробілом, записуємо змінну `слово` як новий елемент внутрішнього масиву `масивРеченнь[i]`:

```
масивРеченнь[i][index] = слово;
```

Після цього збільшуємо змінну `index` (яка відповідає порядковому номеру елементу внутрішнього масиву) на одиницю та витираємо всі символи в змінній `слово`:

```
index++;
слово = "";
```

Оскільки після останнього слова в реченні немає пробілу, ми не потрапимо у блок `else` умовного оператора `if`, де додається новий елемент в масив, і, відповідно, останнє слово речення не буде записане до масиву. Щоб врахувати останні слова речень, ми створили змінну `останнєСлово`:

```
String^ останнєСлово = "";
```

В дану змінну кожного разу записується поточне значення змінної `слово`, якщо символ не є пробілом:

```
if (літера != ' ')
{
	слово += літера;
	останнєСлово = слово;
}
```

По завершенні роботи циклу `for each` в змінній `останнєСлово` зберігатиметься останнє слово речення, яке не було додане до масиву. Тепер нам лише потрібно додати це останнє слово у масив:

```
//Оскільки після останнього слова у реченні немає пробілу, записуємо його окремо після
роботи циклу
масивРеченнь[i][index] = останнєСлово;
```

Надалі виводимо на екран речення після виконання всіх змін та кількість слів у даному реченні:

```
Console::WriteLine(L"\nВиправлене речення:");
Console::WriteLine(речення);
Console::WriteLine(L"Кількість слів у реченні = {0}\n",count);
```

Тепер залишилось лише вивести сформований масив масивів на екран. Це потрібно робити за допомогою двох циклів `for`, один з яких вкладений в інший.

```
//Виводимо резульуючий масив масивів слів за допомогою індексації
Console::WriteLine(L"\nСформований масив з речень");
for (size_t i = 0; i < масивРеченнь->Length; i++)
{
	for (size_t j = 0; j < масивРеченнь[i]->Length; j++)
	{
```

```
			Console::Write(L" S({0},{1})={2}\t", i, j, масивРеченнь[i][j]);
		}
		Console::WriteLine();
}
```

Для порівняння коду ми також додали виведення елементів зубчастого масиву за допомогою циклів for each. В даному циклі звернення до елементів набору перераховуваних значень (в нашому випадку елементів масиву) здійснюється не через індекс, а через *змінну-ідентифікатор*.

```
Console::WriteLine();
//Виводимо результуючий масив масивів слів за допомогою циклу for each
for each (array<String^>^ речення in масивРеченнь)
{
		for each (String^ слово in речення)
		{
			Console::Write(L" {0}\t", слово);
		}
		Console::WriteLine();
}
```

При застосуванні циклу for each потрібно пам'ятати, що ідентифікатор не дозволяє змінювати елементи масиву, а лише дає можливість зчитувати їх значення. Для зміни елементів масиву у циклі for each ідентифікатор має бути *відстежуваним посиланням*, яке позначається символом %. Для можливості зміни елементів масиву в вище наведеному циклі for each потрібно внести наступну змінну у заголовок внутрішнього циклу:

```
for each (String^ %слово in речення)
```

Результатом роботи даної програми буде наступний текст в консольному вікні.

```
Введіть розмірність зовнішнього масиву
4
Добрим словом мур проб'єш, а лихим і в двері не ввійдеш.

Виправлене речення:
добрим словом мур проб'єш а лихим і в двері не ввійдеш
Кількість слів у реченні = 11

Хто мови своєї цурається, хай сам себе стидається.

Виправлене речення:
хто мови своєї цурається хай сам себе стидається
Кількість слів у реченні = 8

Птицю пізнати по пір'ю, а людину по мові.

Виправлене речення:
```

```
птицю пізнати по пір'ю а людину по мові
Кількість слів у реченні = 8

Рідна мова - не полова: її за вітром не розвієш.

Виправлене речення:
рідна мова не полова її за вітром не розвієш
Кількість слів у реченні = 9


Сформований масив з речень
 S(0,0)=добрим   S(0,1)=словом   S(0,2)=мур      S(0,3)=проб'єш  S(0,4)=а        S(0,5)=лихим
S(0,6)=і        S(0,7)=в        S(0,8)=двері    S(0,9)=не       S(0,10)=ввійдеш
 S(1,0)=хто        S(1,1)=мови       S(1,2)=своєї      S(1,3)=цурається           S(1,4)=хай
S(1,5)=сам      S(1,6)=себе       S(1,7)=стидається
 S(2,0)=птицю    S(2,1)=пізнати  S(2,2)=по       S(2,3)=пір'ю    S(2,4)=а        S(2,5)=людину
S(2,6)=по       S(2,7)=мові
 S(3,0)=рідна     S(3,1)=мова      S(3,2)=не        S(3,3)=полова    S(3,4)=її        S(3,5)=за
S(3,6)=вітром   S(3,7)=не       S(3,8)=розвієш

 добрим  словом  мур     проб'єш        а       лихим   і       в       двері   не      ввійдеш
 хто     мови    своєї   цурається      хай     сам     себе    стидається
 птицю   пізнати          по      пір'ю   а       людину  по      мові
 рідна   мова    не      полова  її      за      вітром  не      розвієш
```

В таблиці кодування Windows для консольних додатків в українській мові відсутні літери «ґ» та «і». Літера ґ зустрічається рідше, тому проблему з нею виявити складніше, а от літера «і» зустрічається в багатьох словах і в консолі вона буде замінена на символ «?». Це призведе до невірної роботи програми, тому що ми символ «?» замінюємо на пробіл, як знак пунктуації. Через це слова з літерою «і» розіб'ються на кілька слів. Щоб таких помилок не виникло, при введенні речень українською мовою літеру «і» потрібно замінювати на латинську літеру «`i`».

Повний текст програмного коду наведений в лістингу 1.2.

Приклад програмної реалізації

Приклад створення консольного додатку CLR.

Необхідно створити консольний додаток, який створює таблицю множення 12x12 в двовимірному масиві.

Спробуємо скористатися багатовимірним масивом в наступному прикладі.

За допомогою наведених нижче операторів створюється двовимірний масив.

```
const int SIZE(12);
array<int, 2>^ multiplicationTable (gcnew array<int, 2>(SIZE, SIZE));
```

Перший рядок визначає константне цілочисельне значення, яке задає кількість елементів в кожному вимірі масиву. Другий рядок визначає масив рангу 2, що складається з 12 рядків і 12 стовпців. У ньому розміщуються значення добутків таблиці множення розміром 12x12.

Значення елементів масиву формуються у вкладеному циклі.

```
for (int i = 0; i < SIZE; i++)
	for (int j = 0; j < SIZE; j++)
		multiplicationTable[i, j] = (i + 1)*(j + 1);
```

Зовнішній цикл перебирає рядки, внутрішній - стовпці. Значення кожного елемента дорівнює добутку індексу рядка на індекс стовпця після того, як кожен з них збільшений на 1. Решта коду функції `main()` займається виключно виведенням на екран. Після виведення заголовка таблиці формується лінія, що позначає шапку таблиці наступним чином.

```
for (int i = 0; i <= SIZE; i++)          // Вивести горизонтальну роздільну лінію
	Console::Write(L"_____");
Console::WriteLine();                    // Вивести новий рядок
```

Кожна ітерація циклу виводить п'ять символів підкреслення. Зверніть увагу на те, що верхню межу лічильника циклу визначено як значення константи SIZE включно, тобто виводиться 13 наборів знаків підкреслення, щоб на додаток до 12 стовпців значень сформувати початковий стовпець заголовків рядків.

У наступному циклі виводиться рядок заголовків стовпців таблиці.

```
Console::Write(L"    |");               // Вивести верхній рядок таблиці
for (int i = 1; i <= SIZE; i++)
	Console::Write(L"{0,3} |", i);
Console::WriteLine();                    // Вивести новий рядок
```

Спочатку перед виведенням заголовків стовпців окремо виводиться група пробілів, тому що це - спеціальний випадок без вихідного значення; дані пробіли

знаходяться над початковим стовпцем заголовків рядків. Кожна мітка стовпця виводиться в циклі, потім іде символ нового рядка, готуючи наступний рядок виводу.

Рядки значень виводяться за допомогою вкладеного циклу.

```
for (int i = 0; i<SIZE; i++)            // Вивести інші рядки
{
        Console::Write(L"{0,3} |", i + 1);
        for (int j = 0; j<SIZE; j++)
                Console::Write(L"{0,3} |", multiplicationTable[i, j]);
        Console::WriteLine();           // Вивести новий рядок
}
```

Зовнішній цикл перебирає рядки, а код в зовнішньому циклі формує виведення всього рядка, включаючи його мітку в лівій частині. Внутрішній цикл виводить значення елементів масиву `multiplicationTable`, відповідних `i`-му рядку, розділяючи їх вертикальними лініями.

Подальший код виводить нижню лінію таблиці.

Даний приклад формує наступне виведення.

```
Таблиця множення розміром 12:
_____________________________________________________________________
    |  1 |  2 |  3 |  4 |  5 |  6 |  7 |  8 |  9 | 10 | 11 | 12 |
____|____|____|____|____|____|____|____|____|____|____|____|____|
  1 |  1 |  2 |  3 |  4 |  5 |  6 |  7 |  8 |  9 | 10 | 11 | 12 |
  2 |  2 |  4 |  6 |  8 | 10 | 12 | 14 | 16 | 18 | 20 | 22 | 24 |
  3 |  3 |  6 |  9 | 12 | 15 | 18 | 21 | 24 | 27 | 30 | 33 | 36 |
  4 |  4 |  8 | 12 | 16 | 20 | 24 | 28 | 32 | 36 | 40 | 44 | 48 |
  5 |  5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 | 50 | 55 | 60 |
  6 |  6 | 12 | 18 | 24 | 30 | 36 | 42 | 48 | 54 | 60 | 66 | 72 |
  7 |  7 | 14 | 21 | 28 | 35 | 42 | 49 | 56 | 63 | 70 | 77 | 84 |
  8 |  8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 | 80 | 88 | 96 |
  9 |  9 | 18 | 27 | 36 | 45 | 54 | 63 | 72 | 81 | 90 | 99 |108 |
 10 | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 |100 |110 |120 |
 11 | 11 | 22 | 33 | 44 | 55 | 66 | 77 | 88 | 99 |110 |121 |132 |
 12 | 12 | 24 | 36 | 48 | 60 | 72 | 84 | 96 |108 |120 |132 |144 |
_____________________________________________________________________
```

Текст програмного коду даного модуля наведений в лістингу 1.1.

Програмний код

Лістинг 1.1

```
// Використання двомірного масиву

// Проект_КПР1_1.cpp: главный файл проекта.

#include "stdafx.h"

using namespace System;

int main(array<System::String ^> ^args)
{
	const int SIZE(12);
	array<int, 2>^ multiplicationTable(gcnew array<int, 2>(SIZE, SIZE));
	for (int i = 0; i < SIZE; i++)
		for (int j = 0; j < SIZE; j++)
			multiplicationTable[i, j] = (i + 1)*(j + 1);
	Console::WriteLine(L"Таблиця множення розміром {0}:", SIZE);
	for (int i = 0; i <= SIZE; i++)          // Вивести горизонтальну роздільну лінію
		Console::Write(L"_____");
	Console::WriteLine();                    // Вивести новий рядок
	Console::Write(L"    |");                // Вивести верхній рядок таблиці
	for (int i = 1; i <= SIZE; i++)
		Console::Write(L"{0,3} |", i);
	Console::WriteLine();                    // Вивести новий рядок
	for (int i = 0; i <= SIZE; i++)          // Вивести горизонтальну роздільну лінію з
                                             // вертикальними рисками
		Console::Write(L"____|");
	Console::WriteLine();                    // Вивести новий рядок
	for (int i = 0; i<SIZE; i++)             // Вивести інші рядки
	{
		Console::Write(L"{0,3} |", i + 1);
		for (int j = 0; j<SIZE; j++)
			Console::Write(L"{0,3} |", multiplicationTable[i, j]);
		Console::WriteLine();                // Вивести новий рядок
	}
	for (int i = 0; i <= SIZE; i++)          // Вивести горизонтальну роздільну лінію
		Console::Write(L"_____");
	Console::WriteLine();                    // Вивести новий рядок
	return 0;
}
```

Лістинг 1.2

```
// Використання зубчастого масиву та робота з рядковими даними

// Проект_КПР1_2.cpp: главный файл проекта.

#include "stdafx.h"

using namespace System;
//Підключення бібліотеки для використання регулярних виразів класів Regex та MatchCollection
using namespace System::Text::RegularExpressions;

int main(array<System::String ^> ^args)
{
	Console::WriteLine(L"Введіть розмірність зовнішнього масиву");
	size_t n = Int32::Parse(Console::ReadLine());
	//Створюємо масив рядкових (текстових) масивів
	array<array<String^>^>^ масивРеченнь = gcnew array<array<String^>^>(n);
	for (size_t i = 0; i < n; i++)
	{
		//Зчитуємо введене речення
		String^ речення = Console::ReadLine();
		//Змінюємо всі літери речення на маленькі
		речення = речення->ToLower();
		//Замінюємо в реченні знаки пунктуації на пробіл за допомогою бібліотеки
регулярних виразів
		речення = Regex::Replace(речення, "[-.?!)(,:;]", " ");
		//Замінюємо в реченні потрійні пробіли на одинарні, якщо такі є
		речення = речення->Replace("   ", " ");
		//Замінюємо в реченні подвійні пробіли на одинарні, якщо такі є
		речення = речення->Replace("  ", " ");
		//Відкидаємо пробіли на початку та в кінці речення
		речення = речення->Trim();
		//Рахуємо кількість пробілів, що залишились
		size_t count(0);
		//Створюємо змінну типу Regex і вказуємо рядок, який будемо шукати (пробіл)
		Regex^ пробіл = gcnew Regex(" ");
		//Створюємо змінну типу MatchCollection і присвоюємо їй результати пошуку
співпадінь
		MatchCollection^ співпадіння = пробіл->Matches(речення);
		//Запам'ятовуємо в змінну count кількість знайдених пробілів в реченні + 1
		count = співпадіння->Count+1;
		//Створюємо масив розмірність якого дорівнює кількості слів у реченні
		масивРеченнь[i] = gcnew array<String^>(count);
		size_t index(0);
		String^ слово = "", ^ останнєСлово = "";
		for each (wchar_t літера in речення)
		{
			if (літера != ' ')
			{
				слово += літера;
				останнєСлово = слово;
			}
			else
			{
				масивРеченнь[i][index] = слово;
				index++;
				слово = "";
			}
		}
```

```
            //Оскільки після останнього слова у реченні немає пробілу, записуємо його
окремо після роботи циклу
            масивРеченнь[i][index] = останнєСлово;
            Console::WriteLine(L"\nВиправлене речення:");
            Console::WriteLine(речення);
            Console::WriteLine(L"Кількість слів у реченні = {0}\n",count);
        }
        //Виводимо результуючий масив масивів слів за допомогою індексації
        Console::WriteLine(L"\nСформований масив з речень");
        for (size_t i = 0; i < масивРеченнь->Length; i++)
        {
            for (size_t j = 0; j < масивРеченнь[i]->Length; j++)
            {
                Console::Write(L" S({0},{1})={2}\t", i, j, масивРеченнь[i][j]);
            }
            Console::WriteLine();
        }
        Console::WriteLine();
        //Виводимо результуючий масив масивів слів за допомогою циклу for each
        for each (array<String^>^ речення in масивРеченнь)
        {
            for each (String^ слово in речення)
            {
                Console::Write(L" {0}\t", слово);
            }
            Console::WriteLine();
        }
    return 0;
}
```

Завдання до комп'ютерного практикуму №1

Варіанти

1. Задано довільний цілочисельний масив ***D(m, n)***. В кожному рядку матриці знайти всі від'ємні елементи, які кратні двом, визначити їх суми, кількості та середні значення. Надрукувати матрицю ***D***, значення та координати знайдених елементів, їх суми, кількості та середні значення.
2. Задано довільний цілочисельний масив ***H(m, n)***. В кожному стовпці матриці визначити мінімальні по модулю елементи та їх координати. Матрицю ***H***, значення та координати мінімальних елементів надрукувати.
3. Задано довільну цілочисельну матрицю ***W(m, n)***. Знайти мінімальні елементи, які кратні двом, та їх координати в парних рядках матриці ***W***. Надрукувати матрицю ***W***, мінімальні елементи та їх координати.
4. Задано довільний цілочисельний масив ***E(m, n)***. В кожному непарному стовпці матриці ***E*** визначити максимальні по модулю елементи та їх координати. Матрицю ***E***, значення та координати максимальних елементів надрукувати.
5. Задано довільну цілочисельну матрицю ***B(m, n)***. Знайти суми, кількості та середні значення від'ємних елементів кратних трьом в кожному непарному рядку матриці. Надрукувати матрицю ***B*** та знайдені значення.
6. Задано довільну цілочисельну матрицю ***S(m, n)***. В кожному парному стовпці матриці ***S*** знайти мінімальні елементи кратні двом та їх координати.. Надрукувати матрицю ***S***, значення та координати мінімальних елементів.
7. Задано довільний цілочисельний масив ***L(m, n)***. В кожному парному стовпці матриці ***L*** визначити суми, кількості та середні значення від'ємних елементів кратних п'яти. Матрицю ***L***, значення та координати знайдених елементів, а також отримані результати надрукувати.
8. Задано довільний цілочисельний масив ***T(m, n)***. В 1-му, 4-му, 7-му і т.д. рядках матриці ***T*** визначити мінімальні елементи кратні трьом та їх координати. Матрицю ***T***, значення та координати мінімальних елементів надрукувати.
9. Задано довільну цілочисельну матрицю ***K(n, n)***. В кожному рядку матриці ***K*** визначити суми, кількості та середні значення додатних елементів кратних чотирьом. Матрицю ***K***, їх суми, кількості та середні значення знайдених елементів надрукувати.
10. В кожному стовпці заданої довільної цілочисельної матриці ***G(m, n)***, знайти значення і координати максимальних елементів кратних трьом. Матрицю ***G***, значення та координати максимальних елементів надрукувати.
11. Задано довільну цілочисельну матрицю ***V(m, n)***. В 1-му, 4-му, 7-му і т.д. стовпцях матриці ***V*** визначити суми, кількості та середні значення від'ємних елементів кратних чотирьом. Матрицю ***V***, отримані суми, кількості та середні значення надрукувати.
12. Задано довільну цілочисельну матрицю ***W(m, n)***. Знайти максимальні елементи кратні двом в непарних рядках матриці ***W*** та їх координати. Надрукувати матрицю ***W***, максимальні елементи та їх координати.
13. В 2-му, 5-му, 8-му і т.д. рядках (окремо) довільної цілочисельної матриці ***F (m, n)*** визначити мінімальні елементи кратні п'яти та їх координати. Надрукувати матрицю ***F***, значення та

координати мінімальних елементів.

14. Задано довільну цілочисельну матрицю ***D(m, n)***. Знайти максимальні елементи кратні чотирьом в парних рядках матриці ***D***. Надрукувати матрицю ***D***, максимальні елементи та їх координати.
15. Задано довільну цілочисельну матрицю ***Q(m, n)***. В кожному рядку матриці ***Q*** визначити суми, кількості та середні значення всіх додатних елементів кратних двом. Матрицю ***Q***, визначені суми, кількості та середні значення надрукувати.
16. Задано довільну цілочисельну матрицю ***W(m, n)***. Знайти максимальні елементи кратні двом в 2-му, 5-му, 8-му і т.д. рядках (окремо) матриці ***W*** та їх координати. Надрукувати матрицю ***W***, максимальні елементи та їх координати.
17. Задано довільну цілочисельну матрицю ***P(m, n)***. В кожному парному стовпці матриці ***P*** знайти суми, кількості та середні значення всіх додатних елементів кратних п'яти. Надрукувати матрицю ***P***, визначені суми, кількості та середні значення.
18. Задано довільну цілочисельну матрицю ***K(n, n)***. В кожному непарному стовпці матриці ***K*** визначити мінімальні елементи кратні чотирьом та їх координати. Матрицю ***K***, значення та координати мінімальних елементів надрукувати.
19. В 2-му, 5-му, 8-му і т.д. стовпцях (окремо) довільної цілочисельної матриці ***A(m, n)*** визначити всі додатні елементи кратні трьом, їх суми, кількості та середні значення. Надрукувати матрицю ***A***, а також знайдені результати для вказаних стовпців.
20. Задано довільну цілочисельну матрицю ***B(n, n)***. В кожному стовпці матриці ***B*** (окремо) визначити суми, кількості та середні значення від'ємних елементів кратних двом. Матрицю ***B*** та знайдені значення надрукувати.
21. В непарних рядках (окремо в кожному) довільної цілочисельної матриці ***X(m, n)*** визначити мінімальні елементи кратні п'яти та їх координати. Надрукувати матрицю ***X***, значення та координати мінімальних елементів.
22. В 2-му, 5-му, 8-му і т.д. рядках (окремо) довільної цілочисельної матриці ***Y(m, n)*** визначити суми, кількості та середні значення від'ємних елементів кратних чотирьом. Матрицю ***Y*** та знайдені значення надрукувати.
23. Задано довільну цілочисельну матрицю ***E(m, n)***. В кожному непарному стовпці матриці ***E*** визначити всі додатні елементи кратні трьом, їх суми, кількості та середні значення. Надрукувати матрицю ***E*** та пораховані значення.
24. Задано довільну цілочисельну матрицю ***V(m, n)***. В 1-му, 4-му, 7-му і т.д. стовпцях матриці ***V*** визначити максимальні по модулю елементи та їх координати. Матрицю ***V***, значення та координати максимальних елементів надрукувати.
25. Задано довільний цілочисельний масив ***T(m, n)***. В 1-му, 4-му, 7-му і т.д. рядках матриці ***T*** визначити мінімальні елементи кратні п'яти та їх координати. Матрицю ***T***, значення та координати мінімальних елементів надрукувати.
26. Задано довільну цілочисельну матрицю ***D(m, n)***. Знайти максимальні по модулю елементи в кожному другому рядку матриці ***D***. Надрукувати матрицю ***D***, максимальні елементи та їх координати.

Контрольні питання

1) Що таке середовище CLR? Як створити консольний додаток CLR?
2) Для чого потрібна функція `WriteLine()` мови C++/CLI? В якому просторі імен вона знаходиться?
3) Для чого потрібна функція `Write()` мови C++/CLI? В якому просторі імен вона знаходиться?
4) Як організувати форматоване виведення текстових рядків за допомогою функцій `Write()` та `WriteLine()`?
5) Для чого використовується специфікація формату у вигляді `{n,w:Axx}` та що вона означає?
6) Наведіть основні специфікатори форматування консольного виведення C++/CLI?
7) Для чого використовуються функції `Console::ReadLine()` та `Console::Read()`?
8) Для чого використовується функція `Console::ReadKey()`?
9) Для чого використовується функція функція `Parse()`?
10) В чому відмінність масивів C++/CLI від масівів «рідного» C++?
11) Як створити масив C++/CLI?
12) Для чого використовується цикл `for each` середовища CLR?
13) Для чого використовується функція `Array::Clear()` та як вона працює?
14) Як задати масив випадкових чисел?
15) Як організувати сортування одномірного масиву?
16) Як організувати сортування даних у двох пов'язаних масивах?
17) Для чого використовується функція `BinarySearch()` та як вона працює?
18) Як задаються багатомірні масиви в C++/CLI? Який максимально можливий ранг масиву?

Комп'ютерний практикум №2

Створення додатку Windows Forms

Мета: вивчити принципи створення нового додатку Windows Forms в Visual C++. Засвоїти методи *компіляції*, *запуску* та *відлагоджування* проекту Windows Forms. Ознайомитись з класами Form, GroupBox, Panel, Lable, TextBox, Button, RichTextBox, їх властивостями та методами. Вивчити оператори перетворення типів при роботі з проектами Windows Forms в Visual C++.

Завдання: створити новий пустий проект CLR, додати в нього форму Windows Forms. Створити шаблон додатку Windows Forms. Додати необхідні елементи інтерфейсу у форму згідно отриманого варіанту завдання.

Загальні вимоги.

1) Створити проект Windows Forms.
2) Додати в нього необхідні елементи керування та програмний код для реалізації поставленої задачі згідно отриманого варіанту завдання
3) Програмний код має бути чітко структурований.
4) Імена об'єктів мають нести сенсові навантаження.
5) Програмний код має супроводжуватись коментарями в тексті програми.

Вимоги до виконання.

1) Створити новий пустий проект в Visual C++ типу CLR ⇒ Пустий проект CLR з ім'ям виду Windows Forms.
2) Додати в проект об'єкт Форма Windows Forms.
3) Виконати негобхідні налаштування властивостей проекту та внести необхідні зміни до програмного коду для можливості компіляції та запуску додатку з пустою формою.
4) Після перевірки запущеного додатку створити новий шаблон з ім'ям Windows Forms в переліку нових проектів Visual C++.
5) Створити новий пустий проект в Visual C++ з використанням нового створеного шаблону проектів Windows Forms з ім'ям виду Прізвище_КПР2.
6) Розробити алгоритм розрахунку залежності згідно свого варіанту.
7) Додати необхідні візуальні елементи інтерфейсу додатку скориставшись об'єктами класів GroupBox, Panel, Lable, TextBox, Button та RichTextBox.

8) Підключити за необхідності бібліотеку математичних функцій до заголовочного файлу форми.
9) Організувати введення вхідних даних за допомогою об'єкту TextBox з поясненнями в об'єктах Lable.
10) Організувати виведення вихідних даних та результатів розрахунку в окремих рядках об'єкту класу RichTextBox.
11) Вивести діалогове вікно з останнім розрахованим значенням шуканої змінної.

Теоретичні відомості

2.6 Технологія Windows Forms

Технологія Windows Forms (Форми Windows) являє собою засіб створення додатків Windows, що виконуються з використанням середовища CLR. Під формою (Form) мається на увазі або звичайне вікно, яке служить основою для вікна програми, або діалогове вікно, в яке можуть додаватися інші елементи управління, що забезпечують взаємодію з користувачем. Середовище розробки Visual C++ поставляється зі стандартним набором, до складу якого входить понад шістдесят різних елементів управління, придатних для використання в формі. Оскільки цих елементів управління так багато, розглядатимемо лише деякі найбільш яскраві з них, чого в принципі достатньо для того, щоб отримати уявлення про способи застосування подібних елементів управління і продовжити вивчення інших самостійно. Багато зі стандартних елементів управління передбачають виконання цілком зрозумілої інтерактивної функції. Наприклад, елементи управління Button дозволяють користувачеві клацати на кнопці, а елементи управління TextBox - вводити текст. Деякі з них є контейнерами (containers), тобто являють собою такі елементи управління, які здатні утримувати інші елементи управління. Наприклад, елемент управління GroupBox може містити такі елементи управління, як елементи управління Button або TextBox. Його обов'язком є просте групування цих елементів управління разом для якої-небудь мети і, в разі потреби, надання для них відповідного напису в інтерфейсі GUI. Крім цього, є також безліч елементів управління сторонніх виробників. Якщо неможливо знайти відповідний елемент керування в середовищі розробки Visual C++, завжди можна спробувати відшукати його серед пропозицій сторонніх виробників.

Форма і всі використовувані з нею елементи управління представляються класом C++/CLI. У кожного такого класу є набір властивостей, які, власне кажучи, і визначають поведінку і зовнішній вигляд того чи іншого елемента управління і навіть самої форми. Наприклад, те, чи повинен елемент управління бути видимим у вікні програми, і те, чи повинен він бути доступний користувачеві

для роботи, визначають саме значення, які встановлюються для відповідних властивостей. Значення для властивостей елементів управління можуть встановлюватися як інтерактивно, при проектуванні інтерфейсу GUI з використанням середовища IDE, так і під час виконання, за рахунок додавання в програму нових функцій або вставки додаткового коду в уже існуючі функції за допомогою вікна редактора коду. Операції, які повинні виконуватися з елементом управління, теж задаються за рахунок визначення в класі відповідних функцій.

При створенні проекту програми Windows Forms створюється вікно програми, засноване на класі Form, і весь необхідний для його відображення код. Після створення проекту програми Windows Forms розробник може вдосконалити його чотирма способами.

- Створити графічний користувальницький інтерфейс інтерактивно за допомогою вкладки Конструктор (Design), що відображається у вікні Редактор (Editor), за рахунок вибору бажаних елементів управління у вікні Панель елементів (Toolbox) і їх розміщення в формі (у міру необхідності він також може створювати і додаткові вікна-форми).
- Змінити властивості елементів управління і форм відповідно до потреб додатка за допомогою вікна Властивості (Properties).
- Створити обробники подій. Для цього досить двічі клацнути на потрібному елементі управління у вкладці Конструктор і встановити існуючу функцію як обробник подій для даного елемента управління за допомогою його вікна Властивості.
- Для задоволення потреб програми змінювати і розширювати ті класи, які створюються автоматично в результаті дій на вкладці Конструктор.

Далі в розділі ви побачите на практиці, як виконуються всі ці операції.

2.7 Створення нового додатку Windows Forms

Для того, щоб створити проект додатка на основі технології Windows Forms, якщо ви до цього не створили шаблон для додатків Windows Forms (див. практичну роботу №4), неодхідно виконати наступну послідовність дій.

1) Вибрати в головному меню пункт Файл ⇒ Створити ⇒ Проект… (File ⇒ New ⇒ Project…).
2) В діалоговому вікні Створення проекту (New Project) в лівій частині вікна обрати Встановлені ⇒ Шаблони ⇒ Visual C++ ⇒ CLR (Installed ⇒ Templates ⇒ Visual C++ ⇒ CLR).
3) В діалоговому вікні Створення проекту (New Project) в центральній частині вікна обрати пункт Пустий проект CLR (Empty Project CLR).

4) Задати ім'я нового проекту, місце збереження нового рішення та ім'я нового рішення і натиснути кнопку OK.
5) Обрати в меню пункт Проект ⇒ Додати новий елемент… (Project ⇒ Add new item…).
6) З'явиться діалогове вікно Додати новий елемент (Add New Item…), в лівій частині якого слід обрати пункт Встановлені ⇒ Visual C++ ⇒ UI (Installed ⇒ Visual C++ ⇒UI).
7) Надалі в центральній частині діалогового вікна Додати новий елемент (Add New Item) обрати пункт Форма Windows Forms (Form Windows Forms), задати ім'я нової форми (файл заголовку з розширенням .h), місце збереження файлу нової форми та по завершенню вибору натиснути кнопку Додати (Add) для додавання нової форми у проект.
8) Додати наступний програмний код у файл форми з розширенням .cpp (за замовчуванням файл MyForm.cpp), який обирається в Оглядачі рішень (Solution Explorer).

```
#include "MyForm.h"

using namespace ProjectCLR; //ім'я вашого проекту

[STAThreadAttribute]
int main(array<System::String ^> ^args)
{
	Application::EnableVisualStyles();
	Application::SetCompatibleTextRenderingDefault(false);
	Application::Run(gcnew MyForm());
	return 0;
}
```

9) Обрати пункт меню Проект ⇒ Властивості: Проект1 (Project ⇒ Project1 Propeties…), після чого з'явиться діалогове вікно Сторінки властивостей Проект1 (Project1 Property Pages).
10) В лівій частині діалогового вікна обираємо пункт Властивості конфігурації ⇒ Компонувальник ⇒ Система (Configuration Properties ⇒ Linker ⇒ System).
11) В правій частині діалогового вікна у полі Підсистема (SubSystem) задаємо Windows (/SUBSYSTEM:WINDOWS) (рис. 2.1).

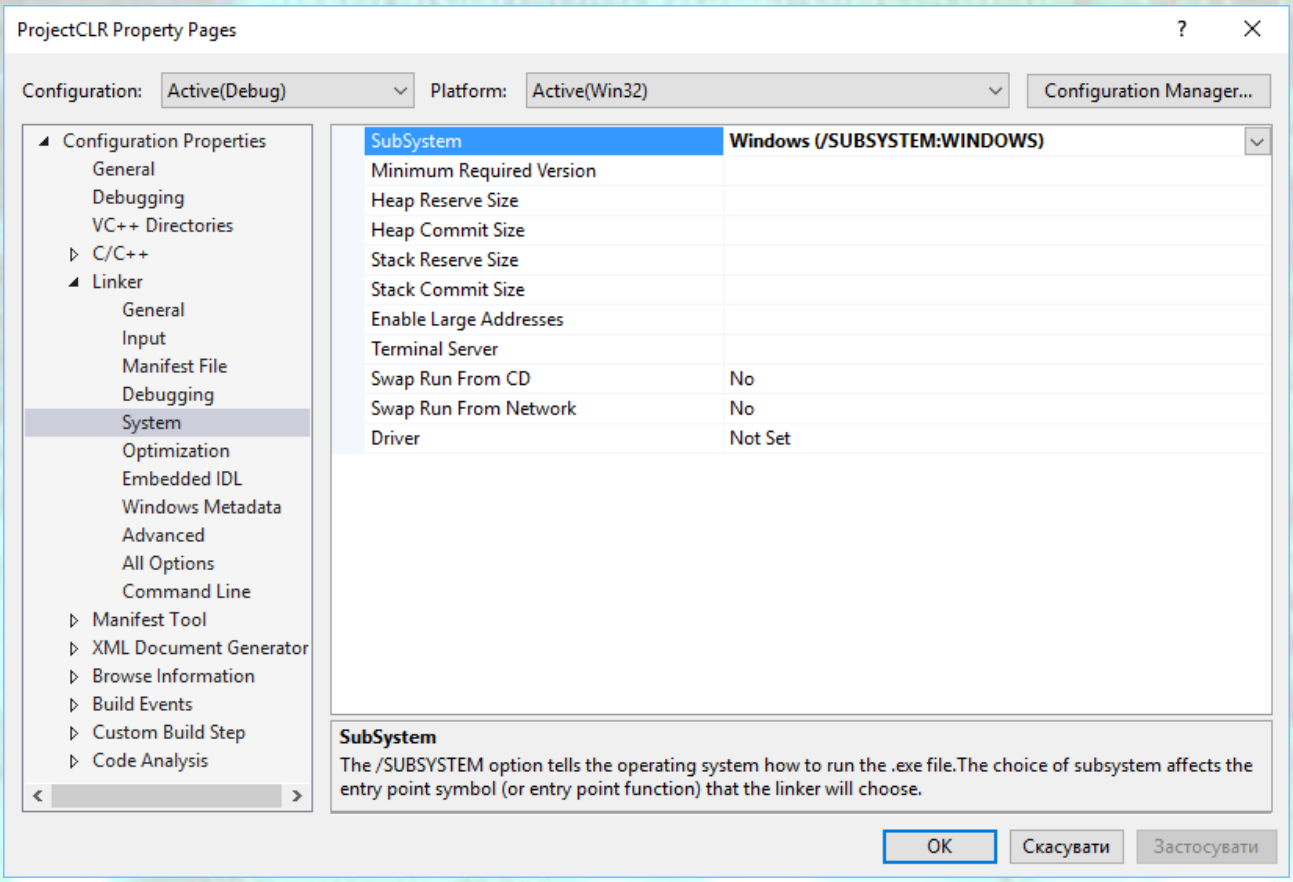

Рисунок 2.1 – Вікно властивостей проекту з вкладкою Компонувальник ⇒ Система

12) В лівій частині діалогового вікна обираємо пункт Властивості конфігурації ⇒ Компонувальник ⇒ Додатково (Configuration properties ⇒ Linker ⇒ Advanced) (рис. 2.2).
13) В правій частині діалогового вікна у полі Точка входу (Entry Point) задаємо main і натискаємо кнопку OK для застосування змін.

Для того, щоб не повторювати кожного разу даний перелік операцій при створенні нового віконного додатку Windows Forms потрібно створений проект додати до шаблону проектів Visual C++.

Для Visual Studio 2017 порядок дій зі створення шаблону проекту дещо змінився у зв'язку з переміщенням пункту меню для створення шаблону на вкладку Проект (Project).

1) Обираємо пункт меню Проект ⇒ Експорт шаблону… (Project ⇒ Export Template…).
2) З'явиться Майстер експорту шаблонів (Export Template Wizard). У вікні майстра Вибір типу шаблону (Choose Template Type) залишаємо все за замовчуванням та натискаємо кнопку Далі (Next) як показано на рис. 2.3.

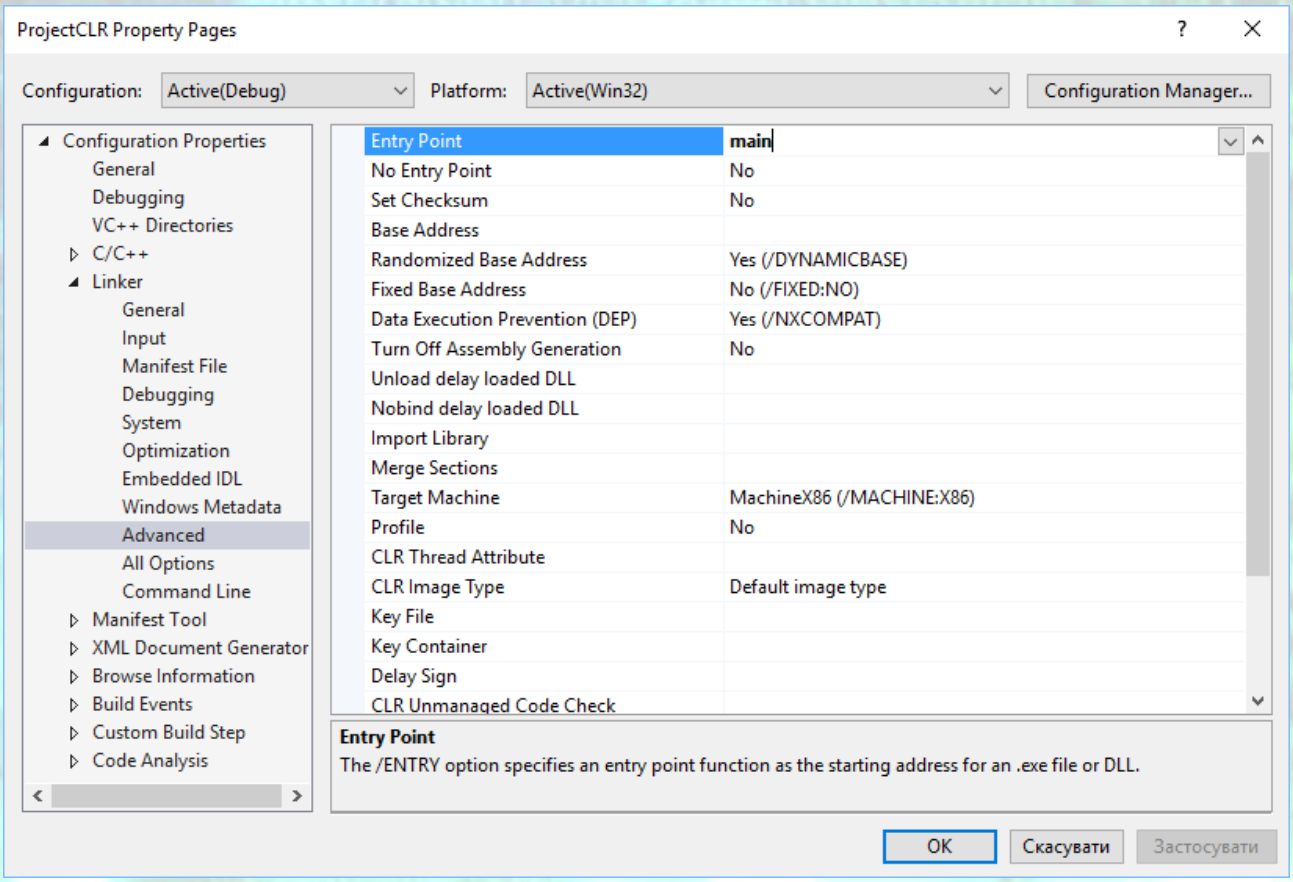

Рисунок 2.2 – Вікно властивостей проекту з вкладкою Компонувальник ⇒ Додатково

3) В другому вікні майстра Вибір параметрів шаблону (Export Template Wizard) додаємо наступне:
   a) у полі Ім’я шаблону: (Template name:) записуємо ProjectCLR;
   b) у полі Опис шаблону: (Template description:) записуємо текст пояснення «Створення віконного додатку Windows Forms (CLR)»;
   c) у полі Зображення значка: (Icon Image:) за бажанням для можливості швидкого знаходження шаблону проекту серед переліку шаблонів Visual C++ додаємо графічне зображення;
   d) перевіряємо вибір опції Автоматично імпортувати шаблон у Visual Studio (Automatically import the template into Visual Studio) та натискаємо кнопку Готово (Finish) як зображено на рис. 2.4.

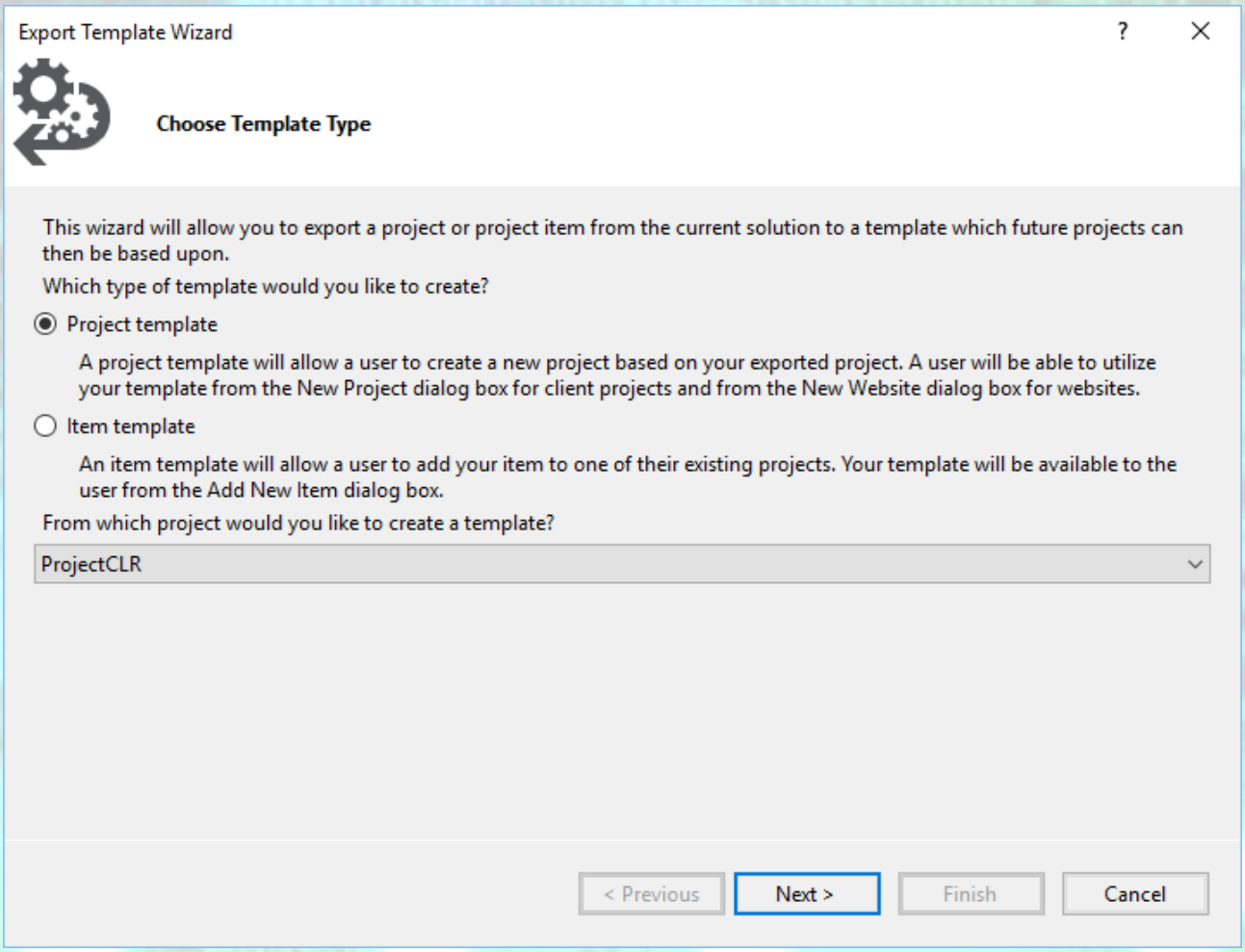

Рисунок 2.3 – Вікно Вибір типу шаблону майстра додавання шаблонів

Після додавання шаблону, для створення нового проекту Windows Forms необхідно буде виконати наступний перелік дій.

1) Обрати в головному меню пункт Файл ⇒ Створити ⇒ Проект… (File ⇒ New ⇒ Project…).
2) В діалоговому вікні Створення проекту (New Project) в лівій частині вікна обрати Встановлені ⇒ Шаблони ⇒ Visual C++ (Installed ⇒ Templates ⇒ Visual C++).

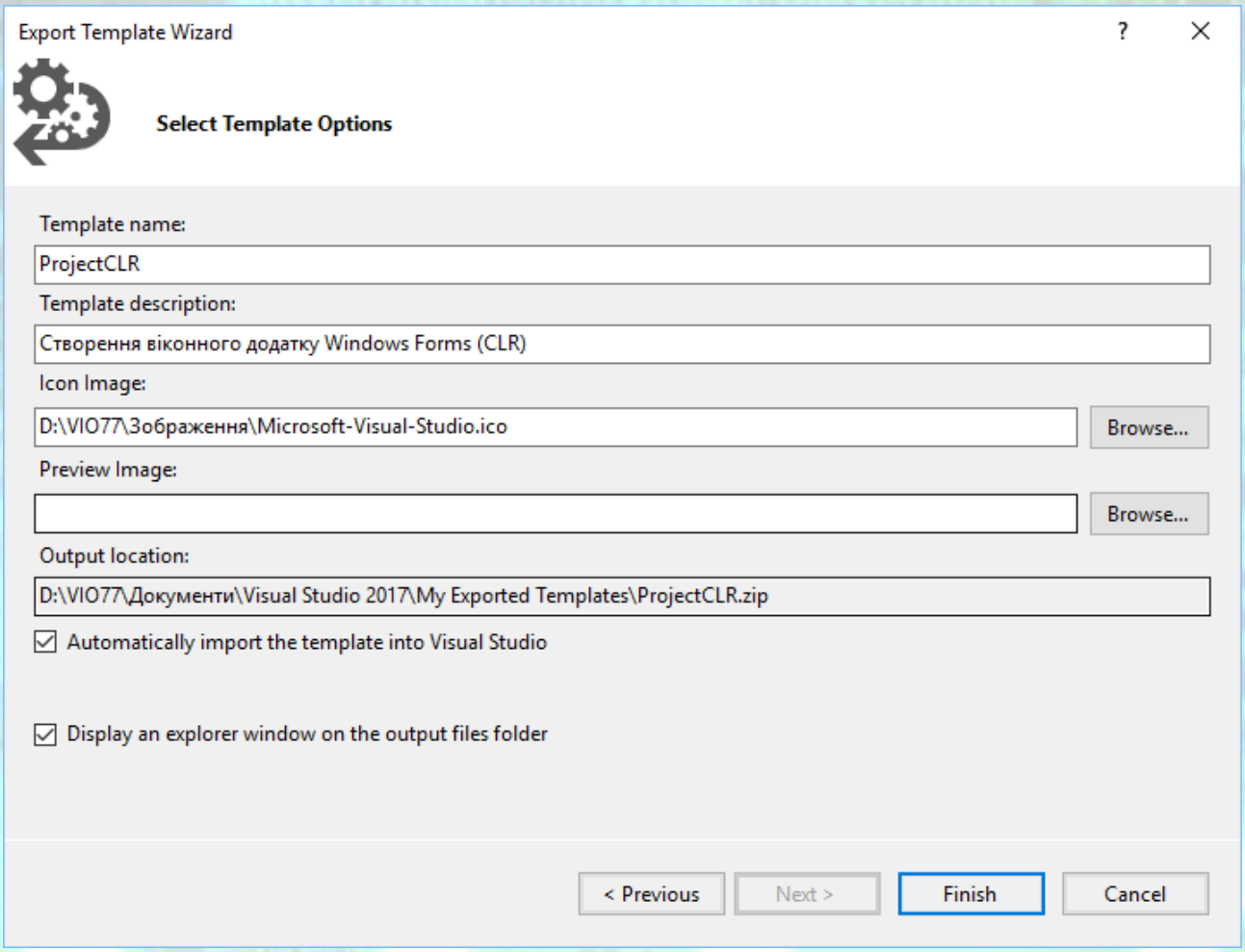

Рисунок 2.4 – Вікно Вибір параметрів шаблону майстра додавання шаблонів

3) В діалоговому вікні Створення проекту (New Project) в центральній частині вікна з'явиться пункт створеного нами шаблону Windows Forms, який і потрібно обрати (рис. 2.5).

Після цього в IDE відкриється новий проект, який вже міститиме пусту форму.

В останніх версіях Visual Studio після створення нового проекту на основі збереженого шаблону виникає помилка, яка пов'язана з відсутністю у файлі шаблону проекту файлів форми MyForm.h та MyForm.cpp. Вирішити цю проблему можна фізичним копіюванням даних файлів у папку проекту (рис. 2.6).

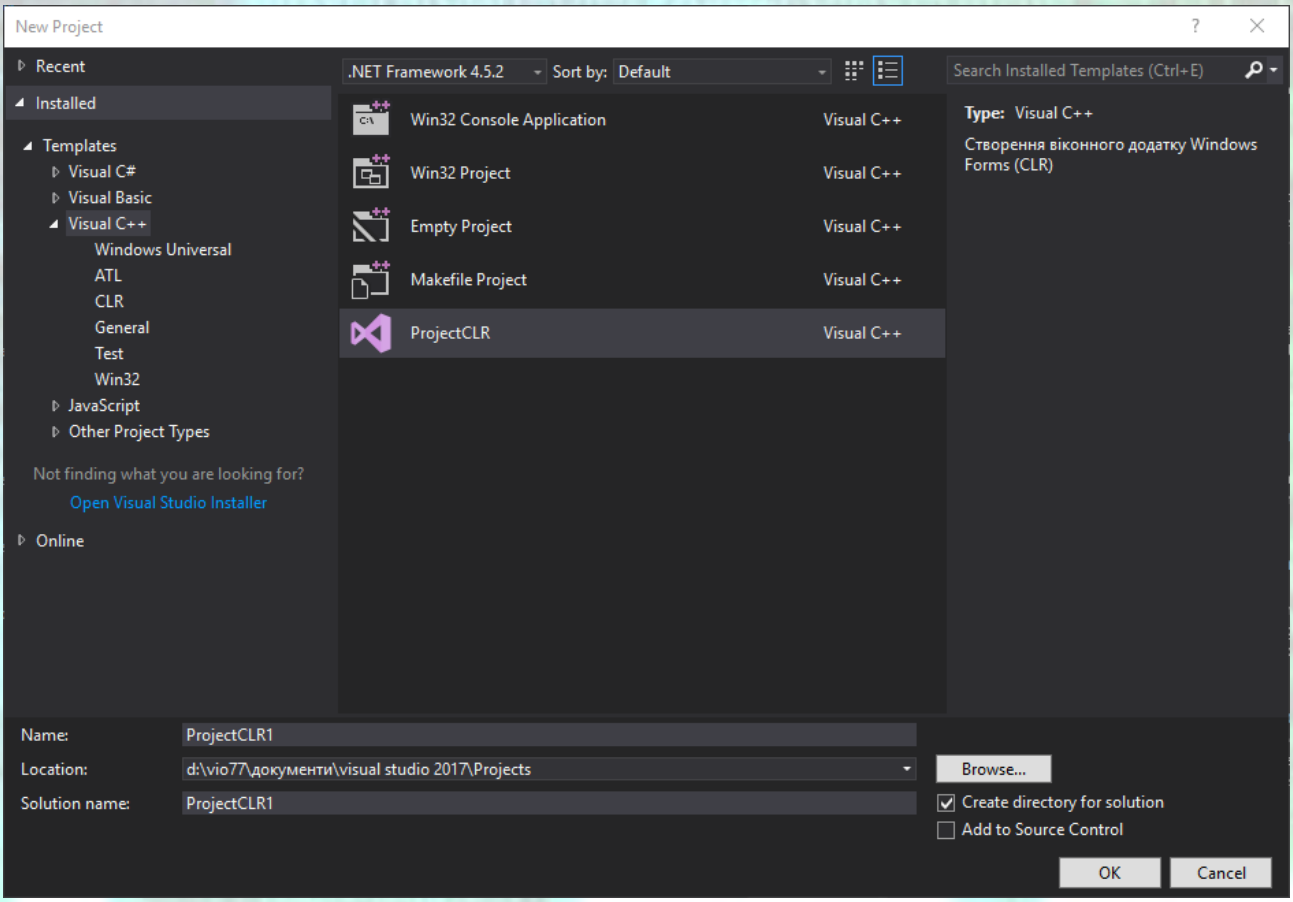

Рисунок 2.5 – Діалогове вікно створення нового проекту з доданим шаблоном `Windows Forms`

При цьому потрібно звернути увагу на те, що імена файлів форми мають повністю співпадати з такими іменами у проекті, а у файлі ресурсів `MyForm.cpp` має бути підключений простір імен, ім'я якого чітко відповідає імені шаблону, на основі якого створений проект:

```
using namespace ProjectCLR; //ім'я вашого шаблону проекту
```

Імена проектам, рішенням та формам потрібно призначати латинськими літерами без пробілів для уникнення в подальшому помилок роботи компілятора.

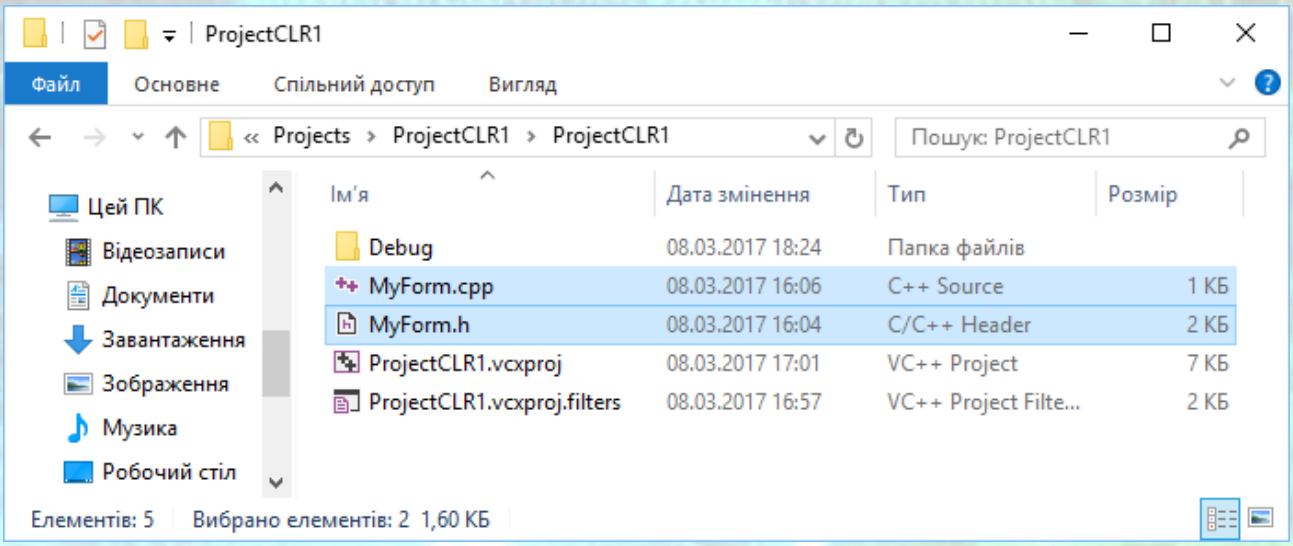

Рисунок 2.6 – Копіювання файлів форми у папку проекту

2.8 Стандартні простори імен бібліотеки .NET

При створенны нового проекту з використанням шаблону Windows Forms в Редакторі (Design) відобразиться графічне представлення вікна додатку – пуста форма (рис. 2.7). В додатках типу Windows Forms графічний інтерфейс користувача (GUI) завжди розробляється графічним способом, тобто додаванням елементів з Панелі елементів (Toolbox) у форму (рис. 2.8).

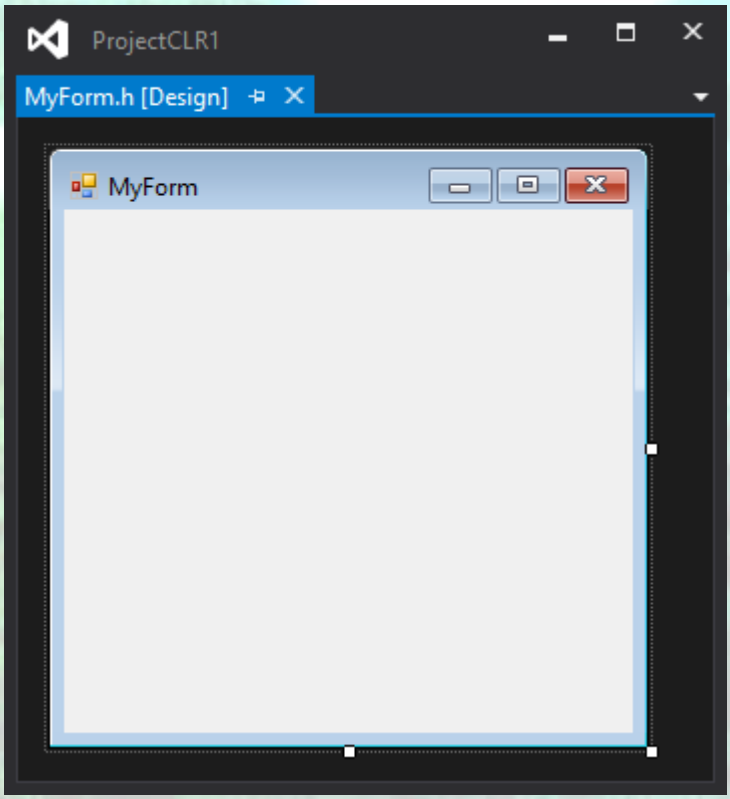

Рисунок 2.7 – Вікно Конструктора (Design) з відкритою формою

При виборі файлу заголовку форми у Оглядачі рішень (Solution Explorer) у Редакторі відобразиться форма. Для перегляду коду файлу форми необхідно натиснути клавішу F7, чи обрати пункт Перейти до коду (View Code) з контекстного меню. Цей код буде визначати клас форми, який за замовчуванням зазвичай має назву Form1 чи MyForm у Visual Studio 2015, 2017. В свою чергу клас форми являє собою вікно додатка, а також створює власний простір імен. Ім'я нового простору імен буде співпадати з ім'ям, яке ми дали проекту, або з ім'ям шаблону, на основі якого створено проект.

```
namespace ProjectCLR {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	// Інша частина коду
}
```

При компіляції буде створена нова збірка, і код цієї збірки буде розміщений у просторі імен *ProjectCLR*, ім'я якого співпадає з іменем проекту чи шаблону, якщо проект створювався на основі шаблону. Простір імен гарантує, що тип в іншій збірці з тим же ім'ям, як і у типа в даній збірці, буде відрізнятися, тому що ім'я кожного типу уточнюється назвою його власного простору імен.

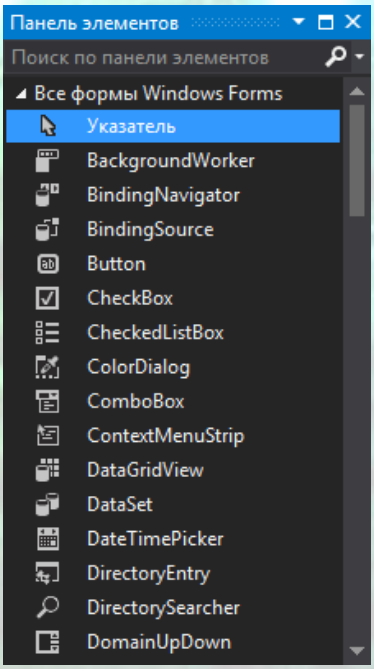

Рисунок 2.8 – Вікно Панель елементів (Toolbox) з набором об'єктів

Крім цього, в коді будуть також присутні шість директив `using` для стандартних просторів імен з бібліотеки `.NET`, що охоплюють найбільш важливі функціональні можливості, які є найбільш імовірними кандидатами на використання в додатку. Короткий опис цих просторів імен наведено в табл. 2.1.

Таблиця 2.1 - Стандартні простори імен з бібліотеки `.NET`

Простір імен	Опис
`System`	Містяться класи, що визначають типи даних, які спільні для всіх програм середовища `CLR`; класи для подій, обробки подій і винятків, а також класи, які підтримують найбільш часто застосовувані функції
`System::ComponentModel`	Містяться класи, які підтримують функціонування компонентів `GUI` в додатку середовища `CLR`
`System::Collections`	Містяться класи колекцій, що дозволяють організувати дані різними способами, а також класи для визначення списків, черг, словників (карт) і стеків
`System::Windows::Forms`	Містяться класи, які підтримують використання в додатку компонентів `Windows Forms`
`System::Data`	Містяться класи, які підтримують технологію `ADO.NET`, яка застосовується для отримання доступу до джерел даних і їх оновлення
`System::Drawing`	Містяться класи, які підтримують основні графічні операції, такі як малювання в формі або компоненті

Клас `MyForm` (чи `Form1`) буде успадкований від класу `Form`, який відноситься до простору імен `System::Windows::Forms`. Клас `Form` може представляти як вікно програми, так і діалогове вікно, а клас `MyForm`, що визначає вікно додатка `ProjectCLR1`, буде успадковувати всі члени класу `Form`.

Розділ в кінці класу `MyForm` буде містити визначення функції `InitializeComponent()`. Ця функція викликається конструктором для установки вікна програми і будь-яких компонентів, що додаються в форму. Коментарі будуть вказувати на те, що змінювати цей розділ коду за допомогою редактора коду неможна, оскільки він буде оновлюватися автоматично по мірі зміни вікна програми інтерактивним чином. Дуже важливо при використанні `Конструктора (Design)` форм не ігнорувати ці коментарі і не намагатися самостійно вносити зміни в створений автоматично код; в іншому випадку в якийсь момент в програмі обов'язково щось піде не так. Звичайно, можна самостійно написати весь код для додатка `Windows Forms` з самого початку, але набагато швидше і надійніше (в плані допущення помилок) використовувати все-таки конструктор форм, що дозволяє налаштовувати графічний користувальницький інтерфейс (GUI) для програми інтерактивним чином. Це,

однак, зовсім не означає, що знати про те, що при цьому відбувається, не обов'язково.

Код для функції InitializeComponent() спочатку буде виглядати наступним чином.

```
private:
        /// <summary>
        /// Обязательная переменная конструктора.
        /// </summary>
        System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
        /// <summary>
        /// Требуемый метод для поддержки конструктора — не изменяйте
        /// содержимое этого метода с помощью редактора кода.
        /// </summary>
        void InitializeComponent(void)
        {
                this->SuspendLayout();
                //
                // MyForm
                //
                this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                this->ClientSize = System::Drawing::Size(284, 261);
                this->Name = L"MyForm";
                this->Text = L"MyForm";
                this->ResumeLayout(false);
        }
```

Спочатку від базового класу тут успадковується член класу MyForm на ім'я components, в обов'язки якого входить стежити за компонентами, які згодом будуть додані в форму. Перший оператор зберігає в змінній components покажчик на об'єкт Container, який представляє колекцію, що зберігає компоненти GUI в списку. Кожен новий об'єкт, який додається до форми за допомогою конструктора форм, буде автоматично додаватися в цей об'єкт Container.

2.9 Зміна властивостей форми

Всі інші оператори в функції InitializeComponent() встановлюють значення для властивостей класу MyForm. Змінювати їх прямо в коді можна, але зручніше для них вибирати інші значення у вікні властивостей форми. Щоб зробити це, поверніться на вкладку конструктора форми MyForm.h у вікні Конструктор (Design) і клацніть на ній правою кнопкою миші та оберіть Властивості (Properties) щоб відкрити відповідне вікно.

На рис. 2.9 властивості форми відображаються за абеткою. Але якщо клацнути на самій першій кнопці у верхній частині вікна Властивості (Properties), то їх можна також розташувати і за призначенням. Варто

переглянути весь перелік властивостей форми, щоб отримати загальне уявлення про доступні можливості.

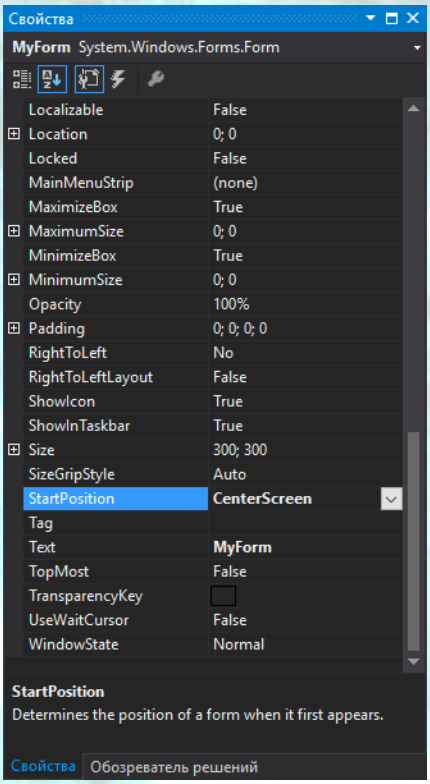

Рисунок 2.9 – Вікно `Властивості` з вкладкою `Properties` форми

Клацання на будь-якій з властивостей призведе до відображення її опису в нижній частині вікна. Змінювати значення тієї чи іншої властивості можна, виділяючи поле, розташоване навпроти неї в правій колонці. Наявність трикутника зліва від властивості означає, що у неї є кілька значень, а тому клацання на цьому знаку призведе до відображення всіх цих значень, після чого їх можна буде змінювати окремо. Відображати властивості в алфавітному порядку (клацанням на відповідній кнопці) зручно в тих випадках, коли відомо ім'я змінюваної властивості, оскільки це спрощує її пошук. Ви можете зробити форму трохи більше, змінивши значення властивості `Size` в групі `Layout` (Компонування) на `500, 350`. Робити форму занадто великою на даному етапі не варто, оскільки це ускладнить роботу з нею після відкриття декількох вікон в IDE, тому робити розмір форми потрібно таким, щоб вам було легко з нею працювати на своєму екрані. Крім цього, можна також змінити значення властивості `Text`, яке в режимі категорій буде відображатися в категорії Appearance (`Зовнішній вигляд`). Це призведе до зміни тексту, відображуваного в рядку заголовка вікна програми. Після цього, якщо ви повернетеся до вікнна `Код`, то побачите, що код функції `InitializeComponent()` змінився і відображає всі ті зміни, які ви внесли в

властивості у вікні Властивості (Properties). З цього випливає, що вікно Властивості є головним засобом для реалізації підтримки компонентів GUI в додатку Windows Forms.

Зверніть увагу на те, що за рахунок зміни значення властивості WindowState ви можете ще зробити і так, щоб при запуску програми вікно додатка відображалося в розгорнутому вигляді. З значенням за умовчанням Normal вікно буде відображатися в тому розмірі, який ви вказали, але зміна значення цієї властивості на Maximized або Minimized за рахунок вибору відповідного варіанту переліку дозволить зробити так, щоб воно відображалося відповідно або в розгорнутому, або згорнутому вигляді.

Також доцільно змінити властивість StartPosition на CenterScreen щоб після запуску вікно додатку розташовувалось в центрі екрану. В іншому разі вікно розташовуватиметься у верхній лівій частині екрану.

2.10 Запуск додатку

Виконання програми починається, як завжди, в функції main(), визначення якої в файлі MyForm.cpp має виглядати наступним чином.

```
int main(array<System::String ^> ^args)
{
	Application::EnableVisualStyles();
	Application::SetCompatibleTextRenderingDefault(false);
	Application::Run(gcnew MyForm());
	return 0;
}
```

Тут функція main() викликає дві статичні функції, визначені в класі Application, визначення якого міститься в просторі імен System::Windows::Forms. Ці статичні функції класу Application є основою кожної програми Windows Forms. Функція EnableVisualStyles(), виклик якої відбувається до виклику функції main(), активізує візуальні стилі для додатка. Функція SetCompatibleTextRenderingDefault() на підставі значення аргументу визначає, як повинен відображатися текст в нових елементах управління. Значення аргументу true вказує, що повинен використовуватися клас Graphics, а значення false задає використання класу TextRenderer. Функція Run() запускає для додатка цикл повідомлень Windows і робить видимим переданий йому як аргумент об'єкт MyForm. Додаток, що виконується за допомогою середовища CLR, в кінцевому підсумку все одно є додатком Windows і тому працює з циклом повідомлень тим же чином, як і всі інші додатки Windows.

Приклад програмної реалізації

Приклад створення проекту Windows Forms з використанням класів Form, GroupBox, Panel, Lable, TextBox, Button, RichTextBox.

Необхідно створити додаток, який розраховуватиме площу колового сегменту. У якості вихідних даних повинні задаватись радіус кола, початковий кут сегменту, кінцевий кут сегменту, крок зміни кута. Розрахунок площі сегменту потрібно організувати в циклі змінюючи кут колового сегменту на заданий крок з кожною ітерацією.

Для створення додатку додайте в нього зазначені компоненти інтерфейсу та присвойте їм потрібні властивості суворо у порядку вказаному в табл. 2.1 для вірного розташування об'єктів в межах форми.

Таблиця 2.1 – Об'єкти додатку Windows Forms та їх властивості

№	Об'єкт	Ім'я	Властивість Text	Інші властивості
1	Forms	MyForm	Площа сегменту	StartPosition: CenterScreen
2	Panel	panel_inp		Dock:Left
3	Panel	panel_out		Dock: Fill
4	GroupBox	groupBox_input	Розрахунок площі колового сегменту	Dock: Fill
5	Button	buttonCalc	Розрахувати площу	Dock: Bottom
6	Label	label_a_beg	Початковий кут a, градусів	Dock: Top
7	TextBox	textBox_a_beg		Dock: Top
8	Label	label_a_end	Кінцевий кут a, градусів	Dock: Top
9	TextBox	textBox_a_end		Dock: Top
10	Label	label_a_step	Крок зміни кута a, градусів	Dock: Top
11	TextBox	textBox_a_step		Dock: Top
12	Label	label_r	Радіус r, мм	Dock: Top
13	TextBox	textBox_r		Dock: Top
14	Button	button_close	Закрити програму	Dock: Bottom
15	RichTextBox	richTextBox_result		Dock: Fill

Після додавання всіх компонентів вікно Конструктора матиме вигляд як зображено на рис. 2.9.

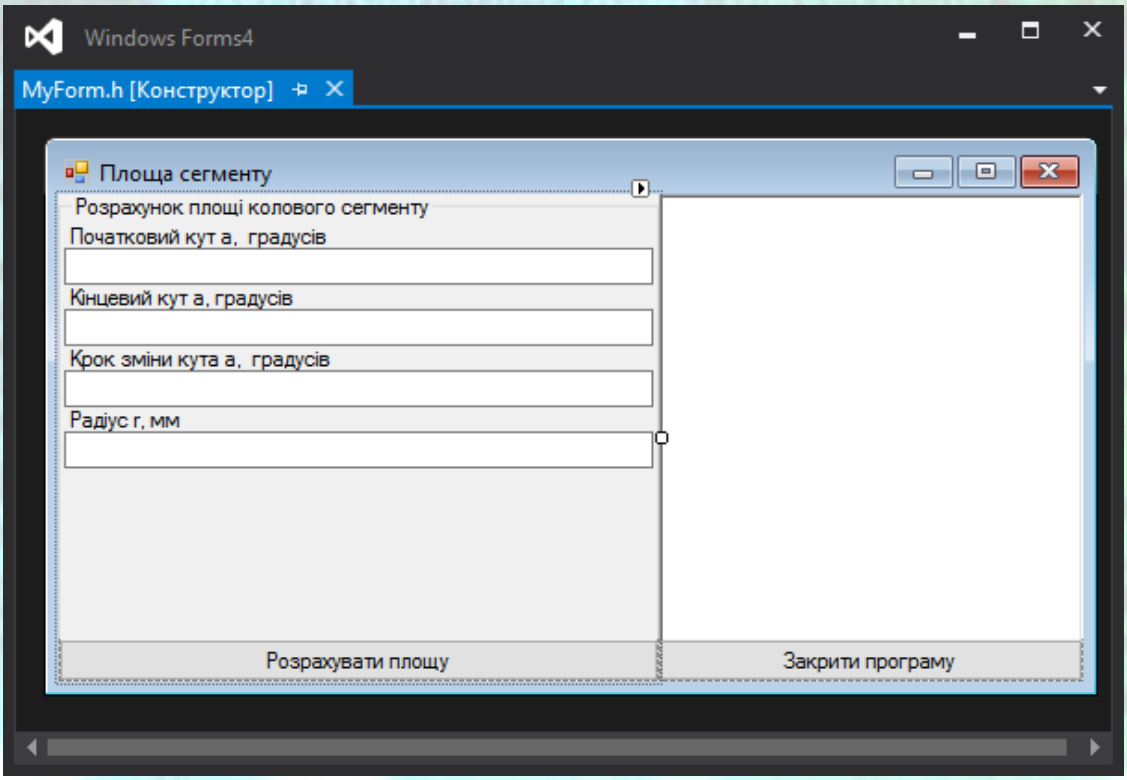

Рисунок 2.10 – Вікно Конструктора форми з доданими елементами інтерфейсу

Двічі клацніть по об'єкту buttonCalc у вікні Конструктора форми. Після цього відкриється вікно Редактора з пустою подією (рис. 2.11).

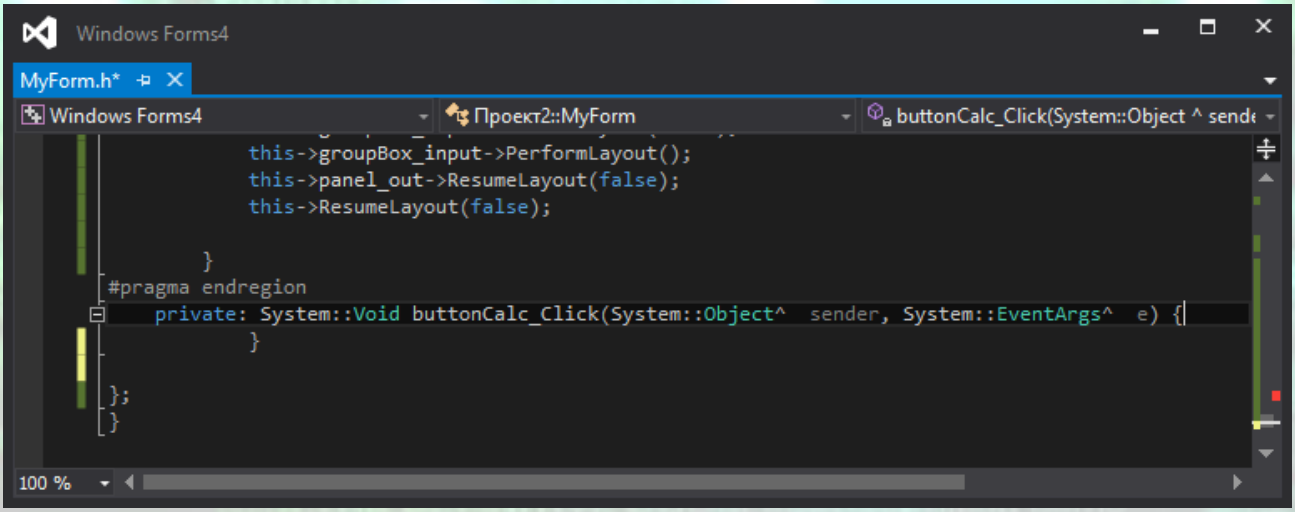

Рисунок 2.11 – Вікно Редактора коду з пустою подією buttonCalc_Click

Надалі впишіть програмний код для розрахунку площі колового сегменту у подію buttonCalc_Click.

```
private: System::Void buttonCalc_Click(System::Object^  sender, System::EventArgs^  e) {
        double r, s;
        int a_beg, a_end, a_step, a;
        //Перетворює рядкове представлення числа в ціле
```

```
        a_beg = int::Parse(textBox_a_beg->Text);
        a_end = int::Parse(textBox_a_end->Text);
        a_step = int::Parse(textBox_a_step->Text);
        r = double::Parse(textBox_r->Text); //Перетворює рядкове представлення числа в дійсне
        a = a_beg;
        //Додає текст, перетворює дійсне значення в рядкове та додає символ кінця рядка
        richTextBox_result->AppendText("Для довжини радіусу r = " + r.ToString("F") +
"\r\n");
        while (a <= a_end)
        {
                s = 0.5*pow(r, 2)*(a - sin(a)); //Розраховує площу колового сегменту
                richTextBox_result->AppendText("при значенні кута a = "+ a.ToString("F") + "
розрахована площа колового сегменту S = " + s.ToString("F") + " мм" + "\r\n");
                a += a_step;
        }
        //Виводить діалогове вікно з повідомленням
        MessageBox::Show("Для довжини радіусу r = " + r.ToString("F") + " при значенні кута a
= " + (a - a_step).ToString("F") + "\r\n" + "розрахована площа колового сегменту S = " +
s.ToString("F") + " мм");
}
```

Двічі клацніть по об'єкту `button_close` у вікні Конструктора форми. Після цього відкриється вікно Редактора з пустою подією `button_close_Click`. Впишіть як реакцію на цю подію функцію закриття програми.

```
private: System::Void button_close_Click(System::Object^  sender, System::EventArgs^  e) {
        Close(); //Закриває вікно програми
}
```

Запустіть проект на виконання та проведіть розрахунки, увівши необхідні вихідні дані. Вікно додатку повинне мати вигляд, як зображено на рис. 2.12.

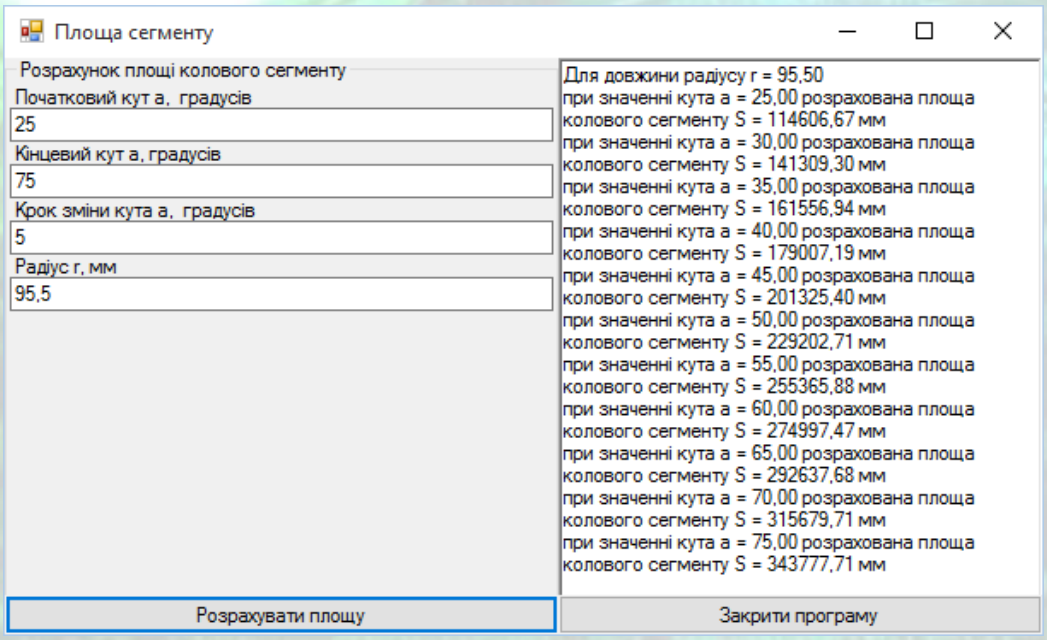

Рисунок 2.12 – Вікно додатку з результатами розрахунку

Окрім цього по завершенню розрахунків на екран виведеться діалогове вікно з результатом розрахунку останньої ітерації (рис. 2.13).

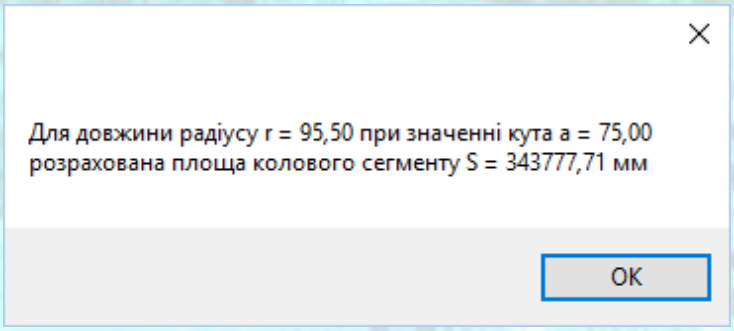

Рисунок 2.13 –Діалогове вікно з результатом розрахунку останньої ітерації

Повний код проекту наведений нижче в лістингах 2.1 та 2.2.

Приклад створення проекту Windows Forms з використанням класів Form, ComboBox, ListBox, Button, RichTextBox.

Необхідно створити додаток, в якому заданий масив масивів з іменеми студентів, що згруповані у внутрішніх масивах в залежності від отриманої оцінки з певної дисципліни. У спадаючому списку ComboBox вибиратиметься оцінка, після чого імена студентів записуватимуться у перелік ListBox. Після натискання кнопки інформація про студентів, що отримали відповідну оцінку додаватиметься в кінець поля RichTextBox, формуючи таким чином певний звіт.

Для створення додатку додайте в нього наступні компоненти інтерфейсу: ComboBox, ListBox, Button, RichTextBox, як показано на рис. 2.14.

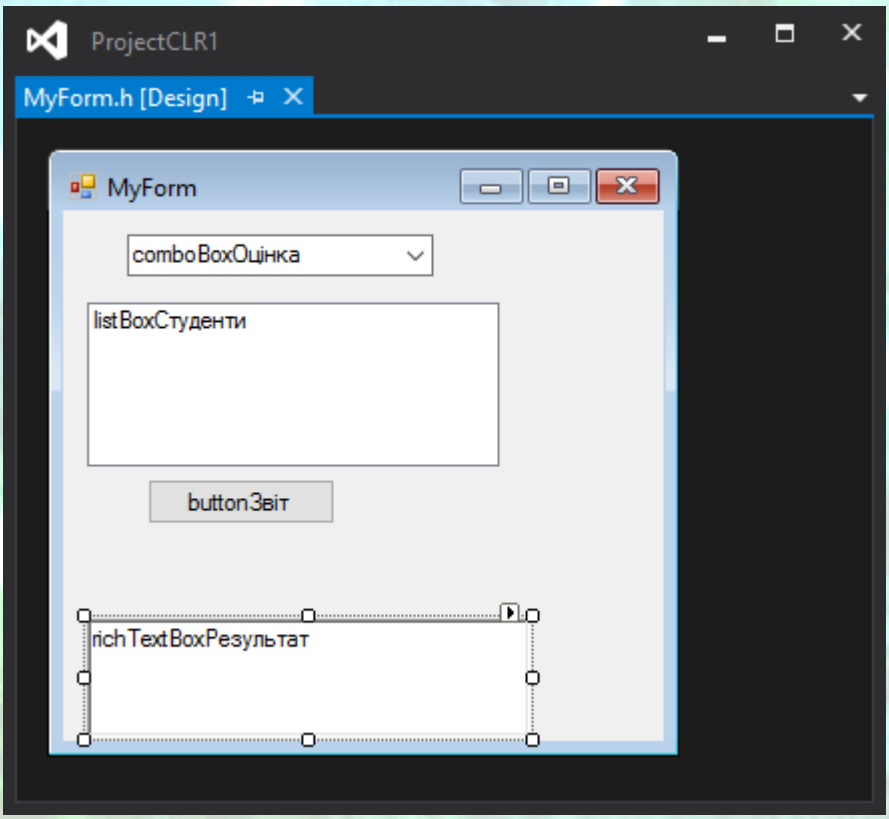

Рисунок 2.14 – Вікно Конструктора форми з доданими елементами інтерфейсу

Присвойте створеним об’єктам класів імена, як вказано на рис. 2.14, для розуміння їх призначення.

Двічі клацніть по об’єкту форми у вікні Конструктора форми. Після цього відкриється вікно Редактора з пустою подією куди необхідно вписати програмний код, що буде виконуватись перед відкриттям вікна програми (функція MyForm_Load()).

```
private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
        студенти = gcnew array<array<String^>^>
        {//Записуємо в масив масивів студентів згідно отриманих оцінок
```

```
            gcnew array<String^>{"Катерина", "Юрко", "Нестор"},        // Оцінка A
                  gcnew array<String^>{"Максим", "Остап", "Богдан", "Одарка"},// Оцінка B
                  gcnew array<String^>{"Валерія", "Ольга", "Тетяна", "Олег"},// Оцінка C
                  gcnew array<String^>{"Яніна", "Назар", "Петро", "Богуслава", "Дмитро"},
                        // Оцінка D
                  gcnew array<String^>{"Орися", "Захар"}                    // Оцінка E
      };
      оцінки = gcnew array<wchar_t> {'A','B','C','D','E'};//Заповнюємо масив оцінок
      for each (wchar_t оцінка  in оцінки)
      {//Записуємо оцінки у спадаючий список comboBoxОцінка
            comboBoxОцінка->Items->Add(оцінка);
      }
      //Вирівнюємо об'єкти по черзі по верхньому краю у формі
      comboBoxОцінка->Dock = System::Windows::Forms::DockStyle::Top;
      listBoxСтуденти->Dock = System::Windows::Forms::DockStyle::Top;
      buttonЗвіт->Dock = System::Windows::Forms::DockStyle::Top;
      //Задаємо ширину текстового поля як у кнопки, яка вже вирівняна у формі
      richTextBoxРезультат->Width = buttonЗвіт->Width;
      //Задаємо положення лівого краю як у кнопки
      richTextBoxРезультат->Left = buttonЗвіт->Left;
      //Задаємо положення верхнього краю з урахуванням розміщених у формі елементів
      richTextBoxРезультат->Top = comboBoxОцінка->Top + listBoxСтуденти->Top
            + buttonЗвіт->Top;
      //Задаємо висоту поля з урахуванням розміщених у формі елементів
      richTextBoxРезультат->Height = this->Height - (comboBoxОцінка->Height
            + listBoxСтуденти->Height + buttonЗвіт->Height + 20);
      //Фіксуємо відстань від країв текстового поля до країв форми щоб вона не змінювалась
      richTextBoxРезультат->Anchor =
            System::Windows::Forms::AnchorStyles(
            System::Windows::Forms::AnchorStyles::Top |
            System::Windows::Forms::AnchorStyles::Left |
            System::Windows::Forms::AnchorStyles::Right |
            System::Windows::Forms::AnchorStyles::Bottom);
      //Додаємо напис на кнопці
      buttonЗвіт->Text = "Звіт";
      //Видаляємо всі записи з текстового поля richTextBoxРезультат
      richTextBoxРезультат->Text = nullptr;
}
```

Перед цією функцією оголошуємо два масиви для того, щоб їх область видимості розповсюджувалась на весь програмний модуль, а не лише на певну функцію.

```
#pragma endregion
      //Задаємо глобальні зміння з областю видимості на рівні програмного модуля
      array<array<String^>^>^ студенти; //Масив студентів згрупованих за оцінками
      array<wchar_t>^ оцінки; //Масив оцінок
```

Це необхідно тому, що дані масиви будуть використовуватись у різних функціях.

У функції `MyForm_Load()` ініціалізуємо два вказані масиви потрібними значеннями:

```
студенти = gcnew array<array<String^>^>
      {//Записуємо в масив масивів студентів згідно отриманих оцінок
        gcnew array<String^>{"Катерина", "Юрко", "Нестор"},        // Оцінка A
```

```
            gcnew array<String^>{"Максим", "Остап", "Богдан", "Одарка"},// Оцінка B
            gcnew array<String^>{"Валерія", "Ольга", "Тетяна", "Олег"},// Оцінка C
            gcnew array<String^>{"Яніна", "Назар", "Петро", "Богуслава", "Дмитро"},// Оцінка D
            gcnew array<String^>{"Орися", "Захар"}                    // Оцінка E
        };
оцінки = gcnew array<wchar_t> {'A','B','C','D','E'};//Заповнюємо масив оцінок
```

Масив студенти є масивам масивів рядкових значень типу String^. Це необхідно для того, щоб можна було записувати різну кількість елементів у його стовпчиках.

Надалі з масиву оцінки у циклі записуємо оцінки до поля зі спадаючим переліком comboBoxОцінка:

```
for each (wchar_t оцінка  in оцінки)
        {//Записуємо оцінки у спадаючий список comboBoxОцінка
                comboBoxОцінка->Items->Add(оцінка);
        }
```

В спадаючомо переліку класу ComboBox значення зберігаються у вигляді рядкового масиву у властивості Items. Для додавання кожного рядка (в нашому випадку оцінки) використовуємо функцію Add().

В подальшому присвоюємо потрібні значення для властивості Dock елементів, яка відповідає за вирівнювання об'єктів у контейнері (в нашому випадку у формі)

```
//Вирівнюємо об'єкти по черзі по верхньому краю у формі
comboBoxОцінка->Dock = System::Windows::Forms::DockStyle::Top;
listBoxСтуденти->Dock = System::Windows::Forms::DockStyle::Top;
buttonЗвіт->Dock = System::Windows::Forms::DockStyle::Top;
```

Для об'єкту richTextBoxРезультат задаємо положення та ширину і висоту в прив'язці до відповідних властивостей інших об'єктів, які вже вирівняні у формі.

```
//Задаємо ширину текстового поля як у кнопки, яка вже вирівняна у формі
richTextBoxРезультат->Width = buttonЗвіт->Width;
//Задаємо положення лівого краю як у кнопки
richTextBoxРезультат->Left = buttonЗвіт->Left;
//Задаємо положення верхнього краю з урахуванням розміщених у формі елементів
richTextBoxРезультат->Top = comboBoxОцінка->Top + listBoxСтуденти->Top + buttonЗвіт->Top;
//Задаємо висоту поля з урахуванням розміщених у формі елементів
richTextBoxРезультат->Height = this->Height - (comboBoxОцінка->Height + listBoxСтудент
        ->Height + buttonЗвіт->Height + 20);
```

За допомогою властивості Anchor фіксуємо відстань країв об'єкту richTextBoxРезультат до верхньої, лівої, правої та нижньої межі форми:

```
//Фіксуємо відстань від країв текстового поля до країв форми щоб вона не змінювалась
richTextBoxРезультат->Anchor = System::Windows::Forms::AnchorStyles(
```

```
System::Windows::Forms::AnchorStyles::Top |System::Windows::Forms::AnchorStyles::Left |
System::Windows::Forms::AnchorStyles::Right | System::Windows::Forms::AnchorStyles::Bottom);
```

Після цього при зміні розміру форми відстані ці мінятись не будуть, тобто об'єкт richTextBoxРезультат змінюватиме свої розміри разом з формою.

Така складна процедура вирівнювання та встановлення розмірів для об'єкту richTextBoxРезультат потрібна, щоб він займав практично всю вільну область форми при зміні її розміру.

Всі ці операції мала замінити властивість Dock = ::DockStyle::Fill, мета якої – заповнення об'єктом усього вільного простору у контейнері, в якому розміщений об'єкт, наприклад у формі. Проте, властивість Dock у Visual Studio працює не коректно і при її застосуванні об'єкт займає всю вільну область без урахування вже розміщених інших об'єктів. Тобто інші об'єкти розташовуються над об'єктом, для якого застосоване значення властивості - Fill і перекривають собою його. Через це використання даної властивості втрачає сенс і потрібно застосовувати властивість Anchor та інші об'єкти – контейнери, такі як Panel та FlowLayoutPanel, в яких можна розміщувати інші об'єкти.

У разі нормальної роботи властивості Dock весь вище наведений програмний код для вирівнювання об'єкту richTextBoxРезультат у формі мав замінити єдиний оператор:

```
richTextBoxРезультат ->Dock = System::Windows::Forms::DockStyle::Fill;
```

Далі на прикінц процедури MyForm_Load() додаємо текст на кнопку

```
//Додаємо напис на кнопці
buttonЗвіт->Text = "Звіт";
```

та очищуємо властивість Text текстового поле richTextBoxРезультат за допомогою присвоєння йому пустого значення – nullptr:

```
//Видаляємо всі записи з текстового поля richTextBoxРезультат
richTextBoxРезультат->Text = nullptr;
```

Для того щоб при обиранні оцінку у спадаючому переліку у відповідному полі з'являлись імена студентів, які цю оцінку отримали, необхідно написати код реакції на подію зміни поточного елементу у молі зі спадаючим списком. За це відповідає обробник події SelectedIndexChanged() об'єкту comboBoxОцінка, який можна знайти на вкладці Events вікна Properties.

```
private: System::Void comboBoxОцінка_SelectedIndexChanged(System::Object^  sender,
System::EventArgs^  e)
{//Процедура викликається при виборі елементу у полі зі спадаючим списком comboBoxОцінка
    //Очищуємо всі елементи у списку
       listBoxСтуденти->Items->Clear();
       for each (String^ студЗОцінкою  in студенти[comboBoxОцінка->SelectedIndex])
       {//В залежності від обраної оцінки в списку comboBoxОцінка додаємо студентів у список
listBoxСтуденти
              listBoxСтуденти->Items->Add(студЗОцінкою);
       }
}
```

В процедурі спочатку очищуємо поле `listBoxСтуденти` за допомогою функції `Items->Clear()`, а потім у циклі `for each` записуємо імена студентів з відповідного стовпчика масиву масивів, який обираємо в залежності від обраного індексу елементу у випадаючому полі з оцінками.

Для додавання реакції на натискання кнопки `buttonЗвіт` достатньо двічі клацнути по ній у вікні редактора форми, щоб з'явилась пуста заготовка обробника події `OnClick`. Або можна обрати необхідну реакцію на подію у вікні `Properties` на вкладці `Events`, як вказано на рис. 2.15.

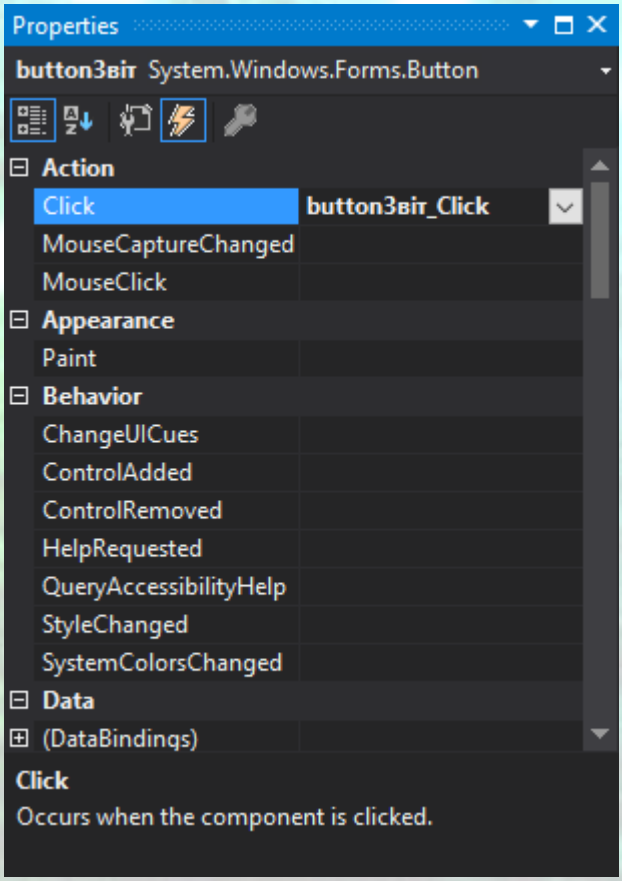

Рисунок 2.15 – Вибір обробника події на вкладці `Events` вікна `Properties`

В цьому вікні можна обирати реакції на всі можливі події для обраного об'єкту, а також присвоювати їм вже існуючі функції. Обробники подій, як і властивості, за замовчуванням згруповані за призначенням чи логічною послідовністю, але за бажанням їх можна відсортувати за абеткою.

Після створення пустого обробника події buttonЗвіт_Click() необхідно додати в нього відповідний програмний код.

```
private: System::Void buttonЗвіт_Click(System::Object^  sender, System::EventArgs^  e) {
	//Формуємо заголовок в полі richTextBoxРезультат для студентів з відповідною оцінкою
	richTextBoxРезультат->AppendText("Оцінку " + comboBoxОцінка->SelectedItem
		+ " отримали:\n");
	for each (String^ студент in listBoxСтуденти->Items)
	{//Додаємо у поле richTextBoxРезультат студентів, які отримали зазначену оцінку
		richTextBoxРезультат->AppendText("    " + студент + ";\n");
	}
	//Додаємо пустий рядок у поле richTextBoxРезультат та переходимо на новий
	richTextBoxРезультат->AppendText("\n");
}
```

У даній процедурі спочатку готуємо заголовок з оцінкою, який додаємо у поле `richTextBoxРезультат`

```
//Формуємо заголовок в полі richTextBoxРезультат для студентів з відповідною оцінкою
richTextBoxРезультат->AppendText("Оцінку " + comboBoxОцінка->SelectedItem + " отримали:\n");
```

а потім в циклі `for each` зчитуємо імена студентів зі списку `listBoxСтуденти`, куди виводяться імена студентів при обранні оцінки у полі `comboBoxОцінка`

```
for each (String^ студент in listBoxСтуденти->Items)
{//Додаємо у поле richTextBoxРезультат студентів, які отримали зазначену оцінку
	richTextBoxРезультат->AppendText("    " + студент + ";\n");
}
```

Для розділення переліку студентів з різними оцінками у полі `richTextBoxРезультат` додаємо пустий рядок

```
//Додаємо пустий рядок у поле richTextBoxРезультат та переходимо на новий
richTextBoxРезультат->AppendText("\n");
```

Вікно програмного модуля з результатами виконання програми приведене на рис. 2.16.

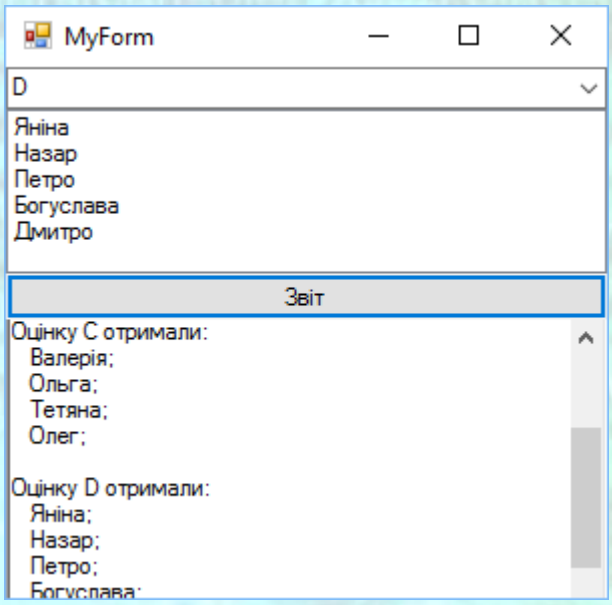

Рисунок 2.16– Вікно додатку з результатами роботи

Повний текст програмного модуля наведений у лістингу 2.3 -2.4.

Програмний код

Лістинг 2.1

```
//Програмний код модуля MyForm.cpp

#include "MyForm.h"

using namespace WindowsForms; //ім’я вашого проекту
//#include <math.h>

[STAThreadAttribute]
int main(array<System::String ^> ^args)
{
	//Активізація візуальних ефектів Windows для створення елементів керування
	Application::EnableVisualStyles();
	Application::SetCompatibleTextRenderingDefault(false);
	//Створення головного вікна та його виконання
	Application::Run(gcnew MyForm());
	return 0;
}
```

Лістинг 2.2

```
//Програмний код заголовку MyForm.h

#pragma once

namespace WindowsForms {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;
	#include <math.h> //Підключення бібліотеки математичних функцій

	/// <summary>
	/// Сводка для MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
		}

	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::Panel^  panel_inp;
	protected:

	private: System::Windows::Forms::Button^  buttonCalc;
	private: System::Windows::Forms::GroupBox^  groupBox_input;
	protected:

	private: System::Windows::Forms::TextBox^  textBox_r;
	private: System::Windows::Forms::Label^  label_r;
	private: System::Windows::Forms::TextBox^  textBox_a_step;
	private: System::Windows::Forms::Label^  label_a_step;
	private: System::Windows::Forms::TextBox^  textBox_a_end;
	private: System::Windows::Forms::Label^  label_a_end;
	private: System::Windows::Forms::TextBox^  textBox_a_beg;
	private: System::Windows::Forms::Label^  label_a_beg;
	private: System::Windows::Forms::Panel^  panel_out;
	private: System::Windows::Forms::RichTextBox^  richTextBox_result;
	private: System::Windows::Forms::Button^  button_close;
```

```
        private:
                /// <summary>
                /// Обязательная переменная конструктора.
                /// </summary>
                System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
                /// <summary>
                /// Требуемый метод для поддержки конструктора — не изменяйте
                /// содержимое этого метода с помощью редактора кода.
                /// </summary>
                void InitializeComponent(void)
                {
                        this->panel_inp = (gcnew System::Windows::Forms::Panel());
                        this->buttonCalc = (gcnew System::Windows::Forms::Button());
                        this->groupBox_input = (gcnew System::Windows::Forms::GroupBox());
                        this->textBox_r = (gcnew System::Windows::Forms::TextBox());
                        this->label_r = (gcnew System::Windows::Forms::Label());
                        this->textBox_a_step = (gcnew System::Windows::Forms::TextBox());
                        this->label_a_step = (gcnew System::Windows::Forms::Label());
                        this->textBox_a_end = (gcnew System::Windows::Forms::TextBox());
                        this->label_a_end = (gcnew System::Windows::Forms::Label());
                        this->textBox_a_beg = (gcnew System::Windows::Forms::TextBox());
                        this->label_a_beg = (gcnew System::Windows::Forms::Label());
                        this->panel_out = (gcnew System::Windows::Forms::Panel());
                        this->button_close = (gcnew System::Windows::Forms::Button());
                        this->richTextBox_result = (gcnew
System::Windows::Forms::RichTextBox());
                        this->panel_inp->SuspendLayout();
                        this->groupBox_input->SuspendLayout();
                        this->panel_out->SuspendLayout();
                        this->SuspendLayout();
                        //
                        // panel_inp
                        //
                        this->panel_inp->Controls->Add(this->buttonCalc);
                        this->panel_inp->Controls->Add(this->groupBox_input);
                        this->panel_inp->Dock = System::Windows::Forms::DockStyle::Left;
                        this->panel_inp->Location = System::Drawing::Point(0, 0);
                        this->panel_inp->Name = L"panel_inp";
                        this->panel_inp->Size = System::Drawing::Size(324, 263);
                        this->panel_inp->TabIndex = 1;
                        //
                        // buttonCalc
                        //
                        this->buttonCalc->Dock = System::Windows::Forms::DockStyle::Bottom;
                        this->buttonCalc->Location = System::Drawing::Point(0, 240);
                        this->buttonCalc->Name = L"buttonCalc";
                        this->buttonCalc->Size = System::Drawing::Size(324, 23);
                        this->buttonCalc->TabIndex = 10;
                        this->buttonCalc->Text = L"Розрахувати площу";
                        this->buttonCalc->UseVisualStyleBackColor = true;
                        this->buttonCalc->Click += gcnew System::EventHandler(this,
&MyForm::buttonCalc_Click);
                        //
                        // groupBox_input
                        //
                        this->groupBox_input->AutoSize = true;
                        this->groupBox_input->Controls->Add(this->textBox_r);
                        this->groupBox_input->Controls->Add(this->label_r);
                        this->groupBox_input->Controls->Add(this->textBox_a_step);
```

```
this->groupBox_input->Controls->Add(this->label_a_step);
this->groupBox_input->Controls->Add(this->textBox_a_end);
this->groupBox_input->Controls->Add(this->label_a_end);
this->groupBox_input->Controls->Add(this->textBox_a_beg);
this->groupBox_input->Controls->Add(this->label_a_beg);
this->groupBox_input->Dock = System::Windows::Forms::DockStyle::Fill;
this->groupBox_input->Location = System::Drawing::Point(0, 0);
this->groupBox_input->Name = L"groupBox_input";
this->groupBox_input->Size = System::Drawing::Size(324, 263);
this->groupBox_input->TabIndex = 1;
this->groupBox_input->TabStop = false;
this->groupBox_input->Text = L"Розрахунок площі колового сегменту";
//
// textBox_r
//
this->textBox_r->Dock = System::Windows::Forms::DockStyle::Top;
this->textBox_r->Location = System::Drawing::Point(3, 128);
this->textBox_r->Name = L"textBox_r";
this->textBox_r->Size = System::Drawing::Size(318, 20);
this->textBox_r->TabIndex = 8;
//
// label_r
//
this->label_r->AutoSize = true;
this->label_r->Dock = System::Windows::Forms::DockStyle::Top;
this->label_r->Location = System::Drawing::Point(3, 115);
this->label_r->Name = L"label_r";
this->label_r->Size = System::Drawing::Size(67, 13);
this->label_r->TabIndex = 7;
this->label_r->Text = L"Радіус r, мм";
//
// textBox_a_step
//
this->textBox_a_step->Dock = System::Windows::Forms::DockStyle::Top;
this->textBox_a_step->Location = System::Drawing::Point(3, 95);
this->textBox_a_step->Name = L"textBox_a_step";
this->textBox_a_step->Size = System::Drawing::Size(318, 20);
this->textBox_a_step->TabIndex = 6;
//
// label_a_step
//
this->label_a_step->AutoSize = true;
this->label_a_step->Dock = System::Windows::Forms::DockStyle::Top;
this->label_a_step->Location = System::Drawing::Point(3, 82);
this->label_a_step->Name = L"label_a_step";
this->label_a_step->Size = System::Drawing::Size(148, 13);
this->label_a_step->TabIndex = 5;
this->label_a_step->Text = L"Крок зміни кута a,  градусів";
//
// textBox_a_end
//
this->textBox_a_end->Dock = System::Windows::Forms::DockStyle::Top;
this->textBox_a_end->Location = System::Drawing::Point(3, 62);
this->textBox_a_end->Name = L"textBox_a_end";
this->textBox_a_end->Size = System::Drawing::Size(318, 20);
this->textBox_a_end->TabIndex = 4;
//
// label_a_end
//
this->label_a_end->AutoSize = true;
this->label_a_end->Dock = System::Windows::Forms::DockStyle::Top;
```

```
                    this->label_a_end->Location = System::Drawing::Point(3, 49);
                    this->label_a_end->Name = L"label_a_end";
                    this->label_a_end->Size = System::Drawing::Size(128, 13);
                    this->label_a_end->TabIndex = 3;
                    this->label_a_end->Text = L"Кінцевий кут a, градусів";
                    this->label_a_end->TextAlign =
System::Drawing::ContentAlignment::BottomCenter;
                    //
                    // textBox_a_beg
                    //
                    this->textBox_a_beg->Dock = System::Windows::Forms::DockStyle::Top;
                    this->textBox_a_beg->Location = System::Drawing::Point(3, 29);
                    this->textBox_a_beg->Name = L"textBox_a_beg";
                    this->textBox_a_beg->Size = System::Drawing::Size(318, 20);
                    this->textBox_a_beg->TabIndex = 2;
                    //
                    // label_a_beg
                    //
                    this->label_a_beg->AutoSize = true;
                    this->label_a_beg->Dock = System::Windows::Forms::DockStyle::Top;
                    this->label_a_beg->Location = System::Drawing::Point(3, 16);
                    this->label_a_beg->Name = L"label_a_beg";
                    this->label_a_beg->Size = System::Drawing::Size(146, 13);
                    this->label_a_beg->TabIndex = 1;
                    this->label_a_beg->Text = L"Початковий кут a,  градусів";
                    //
                    // panel_out
                    //
                    this->panel_out->Controls->Add(this->button_close);
                    this->panel_out->Controls->Add(this->richTextBox_result);
                    this->panel_out->Dock = System::Windows::Forms::DockStyle::Fill;
                    this->panel_out->Location = System::Drawing::Point(324, 0);
                    this->panel_out->Name = L"panel_out";
                    this->panel_out->Size = System::Drawing::Size(229, 263);
                    this->panel_out->TabIndex = 2;
                    //
                    // button_close
                    //
                    this->button_close->Dock = System::Windows::Forms::DockStyle::Bottom;
                    this->button_close->Location = System::Drawing::Point(0, 240);
                    this->button_close->Name = L"button_close";
                    this->button_close->Size = System::Drawing::Size(229, 23);
                    this->button_close->TabIndex = 1;
                    this->button_close->Text = L"Закрити програму";
                    this->button_close->UseVisualStyleBackColor = true;
                    this->button_close->Click += gcnew System::EventHandler(this,
&MyForm::button1_Click);
                    //
                    // richTextBox_result
                    //
                    this->richTextBox_result->Dock =
System::Windows::Forms::DockStyle::Fill;
                    this->richTextBox_result->Location = System::Drawing::Point(0, 0);
                    this->richTextBox_result->Name = L"richTextBox_result";
                    this->richTextBox_result->Size = System::Drawing::Size(229, 263);
                    this->richTextBox_result->TabIndex = 0;
                    this->richTextBox_result->Text = L"";
                    //
                    // MyForm
                    //
                    this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
```

```
                        this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                        this->ClientSize = System::Drawing::Size(553, 263);
                        this->Controls->Add(this->panel_out);
                        this->Controls->Add(this->panel_inp);
                        this->Name = L"MyForm";
                        this->StartPosition =
System::Windows::Forms::FormStartPosition::CenterScreen;
                        this->Text = L"Площа сегменту";
                        this->panel_inp->ResumeLayout(false);
                        this->panel_inp->PerformLayout();
                        this->groupBox_input->ResumeLayout(false);
                        this->groupBox_input->PerformLayout();
                        this->panel_out->ResumeLayout(false);
                        this->ResumeLayout(false);

                }
#pragma endregion
        private: System::Void buttonCalc_Click(System::Object^  sender, System::EventArgs^
e) {
                double r, s;
                int a_beg, a_end, a_step, a;
                a_beg = int::Parse(textBox_a_beg->Text); //Перетворює рядкове представлення
числа в ціле
                a_end = int::Parse(textBox_a_end->Text);
                a_step = int::Parse(textBox_a_step->Text);
                r = double::Parse(textBox_r->Text); //Перетворює рядкове представлення числа в
дійсне
                a = a_beg;
                //Додає текст, перетворює дійсне значення в рядкове та додає символ кінця
        рядка
                richTextBox_result->AppendText("Для довжини радіусу r = " + r.ToString("F") +
"\r\n");
                while (a <= a_end)
                {
                        s = 0.5*pow(r, 2)*(a - sin(a)); //Розраховує площу колового сегменту
                        richTextBox_result->AppendText("при значенні кута a = "+
a.ToString("F") + " розрахована площа колового сегменту S = " + s.ToString("F") + " мм" +
"\r\n");
                        a += a_step;
                }
                //Виводить діалогове вікно з повідомленням
                MessageBox::Show("Для довжини радіусу r = " + r.ToString("F") + " при значенні
кута a = " + (a - a_step).ToString("F") + "\r\n" + "розрахована площа колового сегменту S =
" + s.ToString("F") + " мм");
        }

private: System::Void button_close_Click(System::Object^  sender, System::EventArgs^  e) {
        Close(); //Закриває вікно програми
}
};
}
```

Лістинг 2.3

```
#include "MyForm.h"

using namespace ProjectCLR; //ім'я вашого проекту

[STAThreadAttribute]
int main(array<System::String ^> ^args)
{
	Application::EnableVisualStyles();
	Application::SetCompatibleTextRenderingDefault(false);
	Application::Run(gcnew MyForm());
	return 0;
}
```

Лістинг 2.4

```
//Програмний код заголовку MyForm.h

#pragma once

namespace ProjectCLR {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Summary for MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: Add the constructor code here
			//
		}

	protected:
		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::Button^  buttonЗвіт;
	protected:
	private: System::Windows::Forms::ListBox^  listBoxСтуденти;
	private: System::Windows::Forms::ComboBox^  comboBoxОцінка;
	private: System::Windows::Forms::RichTextBox^  richTextBoxРезультат;
	private:
		/// <summary>
		/// Required designer variable.
		/// </summary>
		System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
		/// <summary>
		/// Required method for Designer support - do not modify
		/// the contents of this method with the code editor.
		/// </summary>
		void InitializeComponent(void)
		{
			this->buttonЗвіт = (gcnew System::Windows::Forms::Button());
			this->listBoxСтуденти = (gcnew System::Windows::Forms::ListBox());
```

```
                    this->comboBoxОцінка = (gcnew System::Windows::Forms::ComboBox());
                    this->richTextBoxРезультат = (gcnew
System::Windows::Forms::RichTextBox());
                    this->SuspendLayout();
                    //
                    // button3віт
                    //
                    this->button3віт->Location = System::Drawing::Point(42, 134);
                    this->button3віт->Name = L"button3віт";
                    this->button3віт->Size = System::Drawing::Size(94, 23);
                    this->button3віт->TabIndex = 11;
                    this->button3віт->Text = L"button3віт";
                    this->button3віт->UseVisualStyleBackColor = true;
                    this->button3віт->Click += gcnew System::EventHandler(this,
&MyForm::button3віт_Click);
                    //
                    // listBoxСтуденти
                    //
                    this->listBoxСтуденти->FormattingEnabled = true;
                    this->listBoxСтуденти->Location = System::Drawing::Point(12, 46);
                    this->listBoxСтуденти->Name = L"listBoxСтуденти";
                    this->listBoxСтуденти->Size = System::Drawing::Size(206, 82);
                    this->listBoxСтуденти->TabIndex = 12;
                    //
                    // comboBoxОцінка
                    //
                    this->comboBoxОцінка->FormattingEnabled = true;
                    this->comboBoxОцінка->Location = System::Drawing::Point(32, 12);
                    this->comboBoxОцінка->Name = L"comboBoxОцінка";
                    this->comboBoxОцінка->Size = System::Drawing::Size(153, 21);
                    this->comboBoxОцінка->TabIndex = 13;
                    this->comboBoxОцінка->Text = L"comboBoxОцінка";
                    this->comboBoxОцінка->SelectedIndexChanged += gcnew
System::EventHandler(this, &MyForm::comboBoxОцінка_SelectedIndexChanged);
                    //
                    // richTextBoxРезультат
                    //
                    this->richTextBoxРезультат->Location = System::Drawing::Point(12, 204);
                    this->richTextBoxРезультат->Name = L"richTextBoxРезультат";
                    this->richTextBoxРезультат->Size = System::Drawing::Size(221, 59);
                    this->richTextBoxРезультат->TabIndex = 10;
                    this->richTextBoxРезультат->Text = L"richTextBoxРезультат";
                    //
                    // MyForm
                    //
                    this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                    this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                    this->ClientSize = System::Drawing::Size(300, 265);
                    this->Controls->Add(this->button3віт);
                    this->Controls->Add(this->listBoxСтуденти);
                    this->Controls->Add(this->comboBoxОцінка);
                    this->Controls->Add(this->richTextBoxРезультат);
                    this->Name = L"MyForm";
                    this->Text = L"MyForm";
                    this->Load += gcnew System::EventHandler(this, &MyForm::MyForm_Load);
                    this->ResumeLayout(false);

                }
#pragma endregion
                //Задаємо глобальні зміння з областю видимості на рівні програмного модуля
                array<array<String^>^>^ студенти; //Масив студентів згрупованих за оцінками
```

```
            array<wchar_t>^ оцінки; //Масив оцінок

private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
       студенти = gcnew array<array<String^>^>
       {//Записуємо в масив масивів студентів згідно отриманих оцінок
              gcnew array<String^>{"Катерина", "Юрко", "Нестор"},          // Оцінка A
              gcnew array<String^>{"Максим", "Остап", "Богдан", "Одарка"}, // Оцінка B
              gcnew array<String^>{"Валерія", "Ольга", "Тетяна", "Олег"},  // Оцінка C
              gcnew array<String^>{"Яніна", "Назар", "Петро", "Богуслава", "Дмитро"},
                     // Оцінка D
              gcnew array<String^>{"Орися", "Захар"}                       // Оцінка E
       };
       оцінки = gcnew array<wchar_t> {'A','B','C','D','E'};//Заповнюємо масив оцінок
       for each (wchar_t оцінка  in оцінки)
       {//Записуємо оцінки у спадаючий список comboBoxОцінка
              comboBoxОцінка->Items->Add(оцінка);
       }
       //Вирівнюємо об'єкти по черзі по верхньому краю у формі
       comboBoxОцінка->Dock = System::Windows::Forms::DockStyle::Top;
       listBoxСтуденти->Dock = System::Windows::Forms::DockStyle::Top;
       buttonЗвіт->Dock = System::Windows::Forms::DockStyle::Top;
       //Задаємо ширину текстового поля як у кнопки, яка вже вирівняна у формі
       richTextBoxРезультат->Width = buttonЗвіт->Width;
       //Задаємо положення лівого краю як у кнопки
       richTextBoxРезультат->Left = buttonЗвіт->Left;
       //Задаємо положення верхнього краю з урахуванням розміщених у формі елементів
       richTextBoxРезультат->Top = comboBoxОцінка->Top + listBoxСтуденти->Top + buttonЗвіт-
>Top;
       //Задаємо висоту поля з урахуванням розміщених у формі елементів
       richTextBoxРезультат->Height = this->Height - (comboBoxОцінка->Height +
listBoxСтуденти->Height + buttonЗвіт->Height + 20);
       //Фіксуємо відстань від країв текстового поля до країв форми щоб вона не змінювалась
       richTextBoxРезультат->Anchor =
System::Windows::Forms::AnchorStyles(System::Windows::Forms::AnchorStyles::Top |
System::Windows::Forms::AnchorStyles::Left | System::Windows::Forms::AnchorStyles::Right |
System::Windows::Forms::AnchorStyles::Bottom);
       //Додаємо напис на кнопці
       buttonЗвіт->Text = "Звіт";
       //Видаляємо всі записи з текстового поля richTextBoxРезультат
       richTextBoxРезультат->Text = nullptr;
}

private: System::Void comboBoxОцінка_SelectedIndexChanged(System::Object^  sender,
System::EventArgs^  e)
{//Процедура викликається при виборі елементу у полі зі спадаючим списком comboBoxОцінка
    //Очищуємо всі елементи у списку
       listBoxСтуденти->Items->Clear();
       for each (String^ студЗОцінкою  in студенти[comboBoxОцінка->SelectedIndex])
       {//В залежності від обраної оцінки в списку comboBoxОцінка додаємо студентів у список
listBoxСтуденти
              listBoxСтуденти->Items->Add(студЗОцінкою);
       }
}

private: System::Void buttonЗвіт_Click(System::Object^  sender, System::EventArgs^  e) {
       //Формуємо заголовок в полі richTextBoxРезультат для студентів з відповідною оцінкою
       richTextBoxРезультат->AppendText("Оцінку " + comboBoxОцінка->SelectedItem + "
отримали:\n");
       for each (String^ студент in listBoxСтуденти->Items)
       {//Додаємо у поле richTextBoxРезультат студентів, які отримали зазначену оцінку
              richTextBoxРезультат->AppendText("    " + студент + ";\n");
```

```
        }
        //Додаємо пустий рядок у поле richTextBoxРезультат та переходимо на новий
        richTextBoxРезультат->AppendText("\n");
}
};
}
```

Завдання до комп'ютерного практикуму №2

Варіанти

1. Розрахувати в'язкість аміаку в інтервалі температур від 12 до 18°С через 1°С за формулою Сатерленда

$$N = N_0 \frac{273{,}15 + C}{T \cdot C} \cdot (T/273{,}15)^n$$

При умові, що множник n визначається так:

$$n = A - B \cdot (t - 273{,}15) \cdot D \cdot 10^{-7} \cdot (t - 273{,}15)^2$$

де *A* = 1,06; *B* = 1,04; *D* = 0; *C* = 503; *N* = 91,6 * 10^{-7} Па*с.

2. Розрахувати ізобарну теплоємність CO_2 в інтервалі температур від 30 до 150°С через 10°С за формулою:

$$C_p = E + F \cdot (T/100) + G \cdot (T/100) + H \cdot (T/100) + N \cdot (T/100)$$

де *E* = 0,81513; *F* = 9,8454· 10^{-2}; *G* = -9,4747· 10^{-3}; *H* = 36,006· 10^{-5}; *N* = -0,0567.

3. Використавши формулу завдання 2 при *E* = 2,0183; *F* = 158,072 * 10^{-2}; *G* = 273,61 * 10^{-3}; *H* = -9380,9 * 10^{-5}; *N* = -1,9986, розрахувати ізобарну теплоємність аміаку в інтервалі температур від 80 до 150°С через 10°С.

4. За умови завдання 2 розрахувати ізохорну теплоємність CO_2. Формула зв'язку має вигляд: $C_v = C_p - R/M$ (*C* – молекулярна маса газу; *R* – універсальна газова стала).

5. За умови завдання 2 розрахувати ізохорну теплоємність аміаку. Формула зв'язку має вигляд: $C_v = C_p - R/M$ (C – молекулярна маса газу; R – універсальна газова стала).

6. Розрахувати в'язкість води в інтервалі температур від 16 до 27°С через 1°С за формулою:

$$N_o = A(B + t)^n,$$

де *A* = 0,59849, *B* = 43, 252, *n* = -1,5423.

7. Розрахувати теплопровідність аміаку в інтервалі температур від 16 до 27°С через 1°С за формулою:

$$L = A \frac{(T_K + c)}{(T + C)} \cdot \frac{T^{\frac{3}{2}}}{T_K^{\frac{3}{2}} M^{\frac{5}{6}}},$$

де A = 4,3643 * 10^{-3}; C = 503; T_K = 239,73 К; *M* – молекулярна маса газу.

8. Розрахувати густину повітря в інтервалі температур від 45 до 54°С за формулою:

$$R = \frac{1{,}293 \cdot P}{(1 + 0{,}00367 \cdot t) \cdot 760}, \text{кг/м}^3$$

де *P* – тиск, який дорівнює 105,2 кПа.

9. Розрахувати швидкість осадження кулеподібної частинки радіусом r ($r = 0{,}2 \cdot 10^{-3}$ м, $r = 0{,}3 \cdot 10^{-3}$ м … $r = 0{,}7 \cdot 10^{-3}$ м) у газі. Використати формулу:

$$W_{oc} = \frac{d^2 \cdot R \cdot g}{18 \cdot M_c},$$

де g = 9,81 м^2/с, R = 1684 кг/м^3; M_c = 5,2 * 10^{-4} Н * с/м^2.

10. Розрахувати час процесу фільтрування суспензії об'ємом V ($V = 50 \cdot 10^{-6}$, $100 \cdot 10^{-6}$, $150 \cdot 10^{-6}$, $200 \cdot 10^{-6}$, $250 \cdot 10^{-6}$ м^3) за формулою:

$$t = \frac{M \cdot r_0 \cdot x_0}{2p} \cdot \frac{V^2}{S^2} + \frac{M \cdot R}{P} \cdot \frac{V}{S},$$

де $M = 5{,}17 \cdot 10^{-9}$ Н * с/м^3; $r = 4{,}15 * 10^{12}$ м$^{-2}$; $R = 2{,}3 * 10^{10}$ м$^{-1}$; $x_0 = 0{,}178$ м^3/м^3; $P = 0{,}6 \cdot 10^5$ Н/м^2; $S = 1$ м.

11. Розрахувати витрату потужності на перемішування для пропелерної мішалки діаметром d = [2,6; 2,8; 3,0; 3,2 м]. Використати формулу

$$N = K_N \cdot n^3 \cdot R_C \cdot d_m^5,$$

де K_N = 0,32; т = 3,96 об/с; R_C = 1200 кг/м^3.

12. Розрахувати коефіцієнт теплопровідності L нітробензолу в інтервалі температур від 40 до 110°С через 10°С. Використати формулу

$$L_t = L_{30} \cdot [1 - E \cdot (t - 30)],$$

де $L_{30} = \mathrm{A}_1 \cdot \mathrm{c} \cdot R \cdot \sqrt{\frac{R}{M}}$, $\mathrm{A}_1 = 4{,}22 \cdot 10^{-2}$; с=1,38· 10^3 Дж/(кг*град); R = 1200 кг/м^3; M = 123; E =1· 10^{-3} град$^{-1}$.

13. Розрахувати значення густини та в'язкості сірчаної кислоти в інтервалі температур від 10 до 80°С через 10°С. Використати формулу

$$R = 1894{,}8 - 0{,}909 \cdot t; \qquad M = 1{,}406 \cdot 10^{-3} + 5{,}0087 \cdot 10^{-1}/t$$

14. Розрахувати опір шару осадка при фільтруванні суспензії з діаметром частинок $d_{\text{ср}}$ = (0,1; 0,15; 0,20; 0,25) * 10^{-3} м за формулою:

$$r_{0_{11}} = A \cdot d_{cp}^B,$$

де A = 9,8373*10; B = -0,3224.

15. Для десяти значень x перевірити, чи є рівняння

$$sin^6 x + cos^6 x = 1 - \frac{3}{4} \cdot sin^2 2x$$

тотожністю.

16. Виконати завдання 16 для формули:

$$\tan x + \tan 2x - \tan 3x = -\tan x \cdot \tan 2x \cdot \tan 3x.$$

17. Виконати завдання 16 для формули:

$$\sin 2nA + \sin 2nB + \sin 2nG = (-1)^{n+1} \cdot 4 \cdot \sin nA \cdot \sin nB \cdot \sin nG,$$

де n = 1, 2, 3, 4…

18. Виконати 5-7 перевірок рівняння

$$\sin A + \sin B + \sin G = 4 \cdot \sin\frac{A+B}{2} \cdot \sin\frac{B+G}{2} \cdot \sin\frac{A+G}{2}$$

на тотожність, якщо *A*, *B* і *G* – кути трикутника.

19. Розрахувати залежність продуктивності вакуумфільтру *W* (кг/(час*м^3)) від величини вакууму *P* (Н/м^2), який визначається від 50 до 100 мм рт. ст. з кроком *P* = 25 мм рт. ст. для трьох розмірів *d* = 0,25; 0,30; 0,35 мм, якщо:

$$W = 7056 \cdot P^{0,5} \cdot d^2$$

20. Розрахувати тиск пару хлористого метилену *P* (та $\ln P$) в інтервалі температур 20…40°С через 5°С, якщо

$$\ln P = -\frac{A}{T} + B \cdot \ln T + C,$$

де *A* = 1995,6; *B* = -3,4426; *T* = *t* + 273; *C* = 8,321; 8,621; 8,921.

21. Розрахувати швидкість *R* хімічної реакції *A* -> *B* в діапазоні температур *t* = 300…400К з кроком 20 К, якщо

$$R = К_0 \cdot \exp\left(-\frac{E}{R} \cdot T\right) \cdot C_0,$$

де $К_0$ = 2,05 * с$^{-1}$; *R* = 1,98725 кал/моль*град; *E* = 20100 кал/моль; C_0 = (2; 4; 6) моль/м^3.

22. Розрахувати, який об'єм займе 10 г азоту при тиску *P* = 1,5; 3,0; 4,5 атм і зміні температури від 200 до 400 К з кроком 10 К.
(Використати формулу 1; *R* = 0,082057 л*атм/моль*град)

23. Розрахувати значення константи швидкості хімічної реакції

$$K = A \cdot \exp\left(\frac{-5,29 \cdot 10^4}{R \cdot T}\right)$$

В інтервалі температур 600 … 800 через 50°С, якщо А = 7,5· 10^{-5}; 8,0· 10^{-5}; 8,5· 10^{-5}.

24. Розрахувати значення енергії Гіббса

$$G = -R \cdot T \cdot \ln K$$

хімічної реакції в інтервалі температур від 400 до 500 К з кроком 20 К, якщо R = 1,3143 Дж/моль*К, К = 9,137; 9,237; 9,337.

25. Розрахувати об'єм кульового сегменту

$$V = \frac{П \cdot h}{6} \cdot (h^2 + 3 \cdot r^2),$$

якщо *h* змінюється від 0,1 до 0,2 м з кроком 0,05 м, для *r* = 0,5; 1,0; 1,5 м.

26. Обчислити та запам'ятати значення:

$$y_{ij} = A_i^{2,2} + 1{,}5 \cdot A_i \cdot B_i \cdot C_j - B_i^{1,5} \cdot \sqrt{A_i + 2B_i}; \qquad A_i = 1{,}8 \cdot \cos^3 x_i; \qquad B_i = e^{2/x_i} + 2{,}6.$$

а потім знайти суму всіх отриманих значень функції y_{ij}.
Прийняти x_i = 2,2; 1,9; 1,7; 1,4; 1,2; C_j = 2,31; 2,86; 3,57.

27. Обчислити та запам'ятати значення $z_i = F(x_i, y_i)$ в n різних точках (x_i, y_i), а потім знайти суму всіх отриманих значень функції. Прийняти $n > 10$; $x_{i+1} = x_i + \delta x$ ($x_0 = 2.8$; $\delta x = 2.1$); $y_{i+1} = y_i + \delta y$ ($y_0 = 1.7$; $\delta y = 0.3$).

$$F(x_i, y_i) = [\ln(x_i^2) - \ln(y_i^2)] \frac{\sqrt[3]{x_i - y_i}}{\operatorname{ctg}(x_i)}.$$

28. Обчислити значення:

$$y_i = a_i^{2b_i} + b_i^{3a_i}; \qquad a_i = e^{2{,}5/x_i} + 3x_i \cdot C; \qquad b_i = \sin^2 x_i + x_i^{0{,}5}.$$

Прийняти x_i = 0,3; 0,7; 0.9; 1,1; 1,2; 1.4; 1,7; 1,8; 1,9; 2.2; 2.5; 2.7; C=2,15;.

29. Розрахувати значення тригонометричних функцій sin(A), cos(A) і ctg(A) при зміні A від 30 до 150° з кроком 15°.

30. Знайти перші десять та обчислити суму перших трьох, п'яти та дев'яти членів ряду, у якому

$$A_n = \frac{2n! - (-1)^n \cdot 2^{n-1}}{2^{n+1} + n!}$$

Контрольні питання

1) Що таке додаток Windows Forms?
2) Як створити новий проект Windows Forms?
3) Як додати нову форму у проект Windows Forms?
4) Як створити шаблон проекту?
5) Що таке Конструктор форми, як додавати нові елементи інтерфейсу у форму?
6) Для чого потрібне вікно Редоктора?
7) Для чого потрібен Оглядач рішень?
8) Як змінити властивості об’єкту? Як додати реакцію на подію для об’єкту?
9) Як організувати перетворення типів при зчитуванні введених даних з об’єкту TextBox?
10) Як організувати перетворення типів при виведенні результатів розрахунку в об’єкт RichTextBox?
11) Як створити діалогове вікно з повідомленням?

Комп'ютерний практикум №3

Створення меню та панелей інструментів

Мета: вивчити прийоми створення меню та панелей інструментів в новому додатку Windows Forms в Visual C++. Засвоїти методи *додавання* та *видалення* компонентів меню та панелей іструментів у проекті Windows Forms, *зміну властивостей* об'єктів головного, контекстного меню та панелей інструментів, *призначення обробників подій* виклику пунктів меню чи команд панелі інструментів. Ознайомитись з елементами управління MenuStrip і ContextMenuStrip та їх об'єктами MenuItem, ComboBox, Separator та TextBox, їх призначенням та властивостями. Ознайомитись з елементом управління ToolStrip та його об'єктами Button, Label, SplitButton, DropDownButton, Separator, ComboBox, TextBox, ProgressBar, їх призначенням та властивостями.

Завдання: створити новий пустий проект CLR, додати в нього форму Windows Forms. Додати у форму елементи головного меню, контекстного меню та панелі інструментів. Додати необхідні елементи інтерфейсу та програмну реалізацію поставленої задачі згідно отриманого варіанту завдання.

Загальні вимоги.

1) Створити проект Windows Forms.
2) Додати в нього необхідні елементи керування та програмний код для реалізації поставленої задачі згідно отриманого варіанту завдання
3) Програмний код має бути чітко структурований.
4) Імена об'єктів мають нести сенсові навантаження.
5) Програмний код має супроводжуватись коментарями в тексті програми.

Вимоги до виконання.

1) Створити новий пустий проект в Visual C++ з використанням нового створеного шаблону проектів Windows Forms з ім'ям виду Прізвище_КПР3.
2) Розробити алгоритм розрахунку залежності згідно свого варіанту.
3) Додати необхідні візуальні елементи інтерфейсу додатку скориставшись об'єктами класів GroupBox, Panel, Lable, TextBox, Button та RichTextBox.
4) Підключити за необхідності бібліотеку математичних функцій до заголовочного файлу форми.

5) Організувати введення вхідних даних за допомогою об'єкту TextBox з поясненнями в об'єктах Lable.
6) Організувати виведення вихідних даних та результатів розрахунку в окремих рядках об'єкту класу RichTextBox.
7) Додати до форми об'єкт головного меню MenuStrip та з його допомогою створити головне меню програми.
8) Додати до форми об'єкт контекстного меню ContextMenuStrip та з його допомогою створити контекстне меню програми.
9) Додати до форми об'єкт панелі управління ToolStrip та з його допомогою створити панель управління програми.
10) Вивести діалогове вікно з останнім розрахованим значенням шуканої змінної.

Теоретичні відомості

3.1 Додавання головного меню

Середовище IDE пропонує стандартний набір елементів управління, які можна інтерактивно додавати в програму за допомогою вікна Конструктора (Design). Натисніть комбінацію клавіш <Ctrl+Alt+X> або виберіть відповідний пункт меню Вид (View), щоб відкрити вікно Панель елементів (Toolbox) зі списком всіх доступних елементів управління (рис. 3.1).

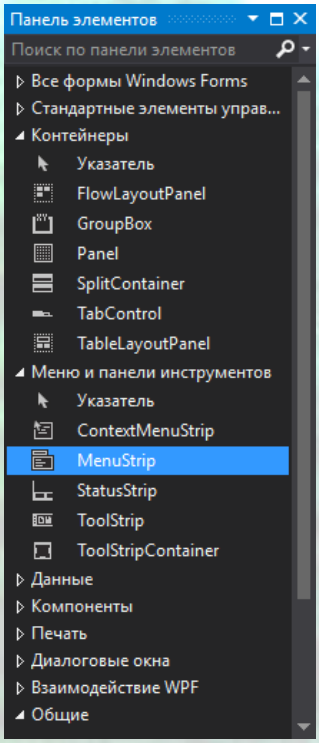

Рисунок 3.1 – Вікно Панель елементів (Toolbox) з набором об'єктів

Як видно на рис. 3.1, перший розділ у вікні Панель елементів називається Всі форми Windows Forms (All Windows Forms) і містить перелік всіх елементів управління, доступних для використання в формі. Клацання на відображеному зліва від заголовка цього розділу трикутнику дозволить розгорнути його, а якщо він розгорнутий, то згорнути його і побачити що він ще також є групою певного типу в списку, який починається з розділу Стандартні елементи управління (Common Controls). Спочатку вам буде зручніше користуватися тільки цією групою, пропонованої на самому початку списку, яка містить всі елементи управління. Після ознайомлення з доступними елементами управління простіше буде згортати групи і залишати розгорнутої тільки якусь одну конкретну групу.

Контейнером для пунктів меню служить смуга меню, тому додайте елемент управління MenuStip в форму, перетягнувши його з розділу Меню і панелі інструментів (Menus & Toolbars) у вікні Панель елементів (Toolbox). Він автоматично розміститься у верхній частині форми прямо під рядком заголовка. У лівому верхньому кутку цього елемента буде відображатися поле вводу з переліком. В ньому безпосередньо можна почати задавити пункти меню, або натиснути на невелику стрілку біля поля вводу та обрати потрібний пункт як показано на рис. 3.2.

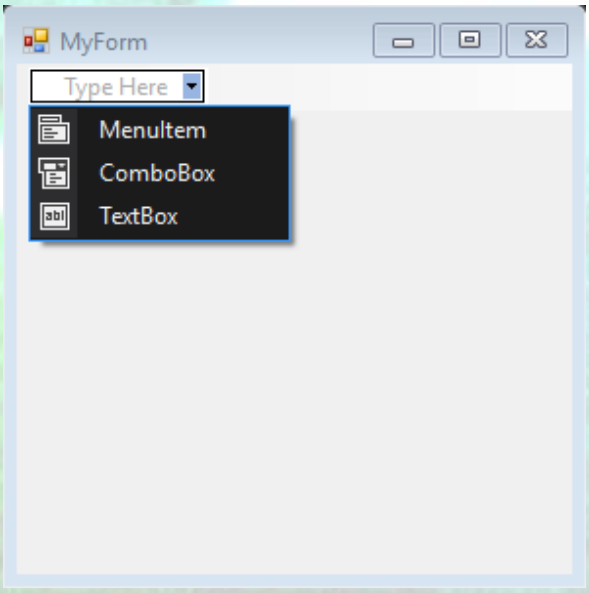

Рисунок 3.2 – Вікно форми з розміщеним елементом MenuStip

Зі спадаючого меню можна обрати який саме елемент ви хочете розмістити в головному меню вашого вікна: пункт меню MenuItem; поле введення зі спадаючим списком ComboBox; текстове поле TextBox. Ці елементи можна безпосередньо розміщувати у смузі меню. Окрім того у спадаючому

меню можна розмістити елемент Separator – це роздільна лінія між пунктами меню (рис. 3.3).

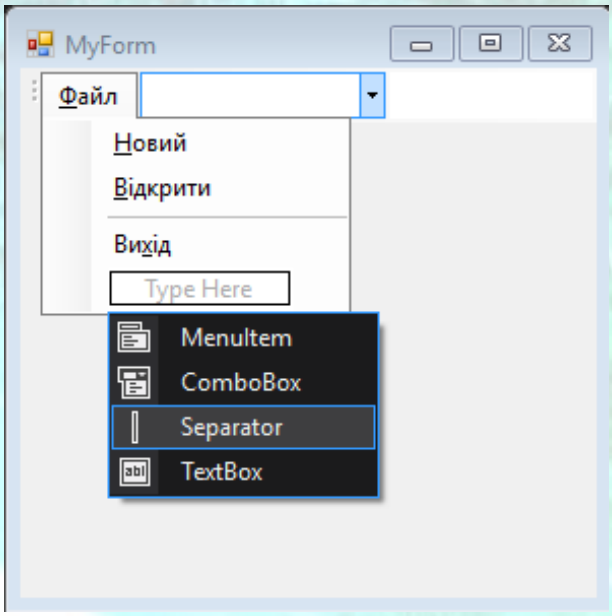

Рисунок 3.3 – Створення меню за допомогою конструктора

На рис. 3.3 у кожному з пунктів меню підкреслена певна літера. Це означає, що при натисканні комбінації клавіш Alt+Літера буде виконуватись зазначений пункт меню. Щоб зробити певну літеру підкресленою в пункті меню потрібно перед цією літерою ввести знак &, наприклад &Новий. Знак & в меню відображатись не буде.

У вікні Властивості (Properties) відображатимуться властивості контейнера меню класу MenuStrip, або властивості елементів меню, які належать до класу MenuItem, в залежності від того, який елемент обраний у Конструкторі чи у вікні властивостей зі спадаючого списку розміщених у додатку елементів. Призначення деяких властивостей для об'єктів MenuItem наведене в таблиці 3.1.

Таблиця 3.1 – Властивості пунктів меню

Простір імен	**Опис**
Name	Ім’я об’єкту
Checked	При значенні True пункт меню відображатиметься відміченим, при значенні False - ні
DisplayStyle	За замовчуванням має значення ImageAndText, що дозволяє відображати у пункті меню як малюнок, так і текст. Може також приймати значення None, Text, Image
Image	Дозволяє обрати малюнок для відображення в пункті меню
Text	Текст, який відображатиметься як ім’я пункту меню
ToolTipText	Дозволяє писати текст спливаючої підказки, який відображатиметься при наведенні вказівника миші на пункт меню

CheckOnClick	Значення True позначати пункт меню натисканням миші або знімати позначку. Значення False за замовчуванням і не дозволяє позначати пункт натисканням миші
Enabled	Значення True робить пункт меню доступним для вибору, значення False – робить його неактивним (відображається сірим кольором)
Visible	Значення True робить пункт меню видимим, значення False приховує його
ShortcutKeys	Дозволяє задавати сполучення гарячих клавіш для виклику пункту меню

3.2 Додавання обробників подій до пунктів меню

В програмі C++/CLI обробники подій є делегатами. *Делегати* - це об'єктно-орієнтовані покажчики на функції, які використовуються для callback-викликів в середовищі CLR фірми Microsoft. Делегат можна пов'язати зі статичною функцією або з нестатичним методом будь-якого класу (єдина умова - збіг сигнатури методу з сигнатурою, зазначеної в описі делегата). Потім пов'язану з делегатом функцію або метод можна викликати, використовуючи стандартний синтаксис виклику функції в C++.

Для додавання реакції на подію до елементу меню необхідно двічі клацнути мишею на потрібному пункті меню у вікні Конструктора (Design). Можна також виділити пункт меню у вікні Конструктора та перейти до вікна Властивості (Properties). У вікні Властивості необхідно натиснути кнопку Events (Події) для відкриття відповідної вкладки. На вкладці Events вікна Властивості потрібно в розділі Action обрати подію Click і двічі клацнути у пустому полі праворуч (рис. 3.4).

Результатом цих дій буде створення пустої функції – обробника події Click, тобто реакції на клацання мишею по даному пункту меню. При цьому відкриється вікно редактору коду з курсором всередині тіла цієї функції.

```
private: System::Void вихідToolStripMenuItem_Click(System::Object^  sender,
System::EventArgs^  e) {
}
```

Тепер після відкритої фігурної дужки можна писати код реакції на дану подію – виклик відповідного пункту меню.

Перший параметр створеної функції вказує на джерело події, яким в даному випадку є об’єкт пункту меню, а другий параметр надає інформацію, яка відноситься до цієї події.

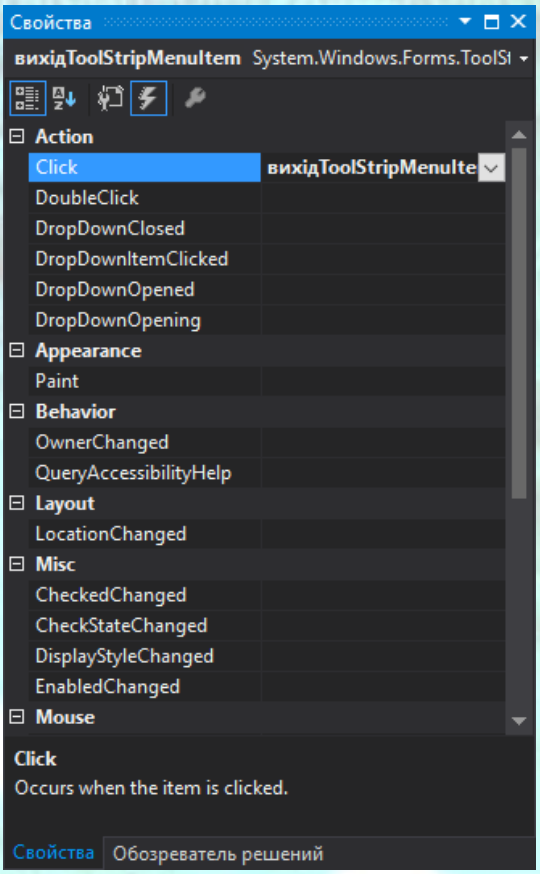

Рисунок 3.4 – Вкладка `Events` вікна `Властивості` з обробником події `Click`

Дана функція автоматично була призначена обробником події для пункту меню `Вихід` за допомогою коду, який міститься у функції `InitializeComponent()` заголовочного файлу форми.

```
// вихідToolStripMenuItem
//
this->вихідToolStripMenuItem->Name = L"вихідToolStripMenuItem";
this->вихідToolStripMenuItem->Size = System::Drawing::Size(152, 22);
this->вихідToolStripMenuItem->Text = L"Ви&хід";
this->вихідToolStripMenuItem->Click += gcnew System::EventHandler(this,
&MyForm::вихідToolStripMenuItem_Click);  //Призначення обробника події
```

Таким же чином можна призначити обробники подій для інших пунктів меню. Також це можна зробити обравши потрібний пункт меню зі спадаючого списку у верхній частині вікна `Властивості` при відкритому вікні конструктора форми (рис. 3.5). Надалі потрібно перейти на вкладку `Events` (`Події`) та обрати необхідну подію, реакцію на яку ми хочемо запрограмувати.

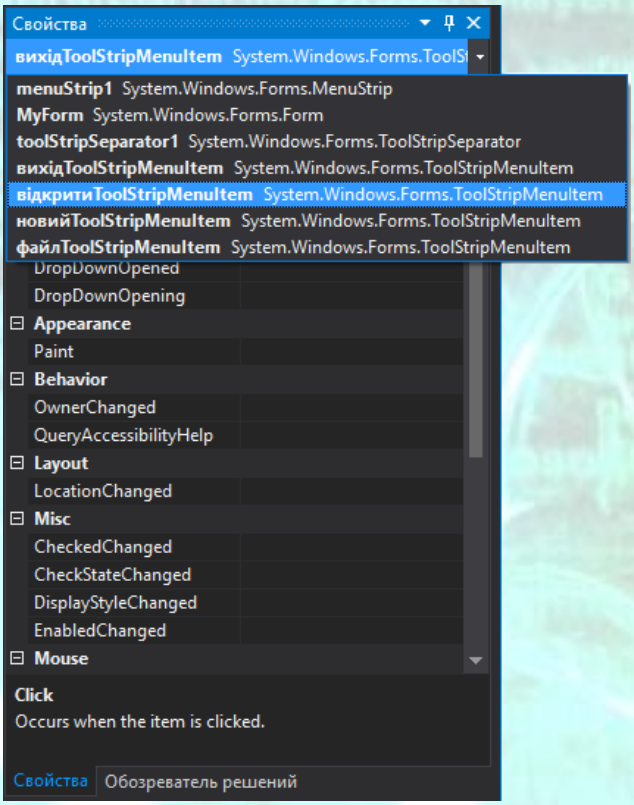

Рисунок 3.5 – Вибір потрібного об’єкту у вікні Властивості

Головне меню може розташовуватись лише у вікні програми і не може розташовуватись всередині окремих об’єктів. Проте форма може мати кілька об’єктів головного меню – об’єктів класу `MenuStrip`. В кожен певний період часу у вікні програми може бути лише одне головне меню, але його можна замінювати на інше у процесі виконання програми. Тому у об’єкта форми є властивість `MainMenuStrip` в якій задається ім’я об’єкту головного меню. Наявність кількох об’єктів головного меню в програмі необхідна в тих випадках, коли склад головного меню має суттєво мінятися в залежності від поточного стану програми. В такій ситуації набагато простіше створити два окремих головних меню та замінювати їх, ніж приховувати певну кількість одних пунктів та відображати певну кількість інших кожного разу, коли це вимагатиме програма.

3.3 Створення контекстного меню

Для створення контекстного меню – меню яке викликається при натисканні правої кнопки миші в межах вікна програми чи якогось елементу інтерфейсу, необхідно у вікні `Панель елементів (Toolbox)` з розділу `Меню і панелі інструментів (Menus & Toolbars)` обрати елемент `ContextMenuStrip` та розмістити його у вікні редактора форми.

Створення пунктів контекстного меню повністю аналогічно створенню пунктів головного меню (рис. 3.6). Після додавання об'єкту класу ContextMenuStrip до вікна проектувальника форми, на формі з'явиться конструктор меню.

На відміну від головного меню, контекстне меню до його виклику на формі не відображається. Тому щоб продовжити створення чи редагування контекстного меню потрібно у нижній частині вікна Конструктора вибрати відповідний об'єкт класу ContextMenuStrip.

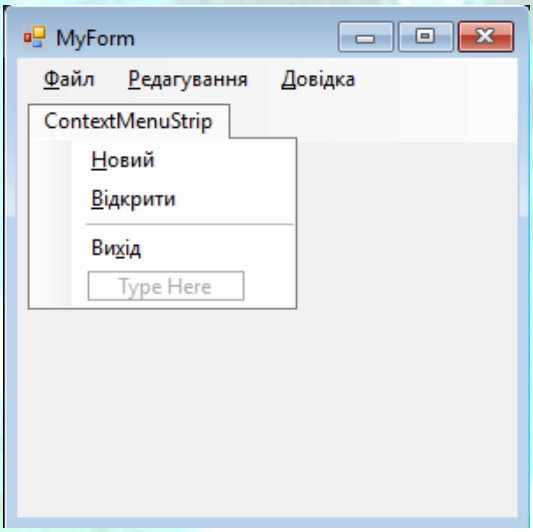

Рисунок 3.6 – Створення контекстного меню у вікні редактора форми

Об'єкти menuStrip1 та contextMenuStrip1 є так званими *невізуальними об'єктами*. Тобто ці об'єкти, як і деякі інші, наприклад об'єкти діалогових вікон, не відображаються у вікні запущеної програми. Так об'єкти menuStrip1 та contextMenuStrip1 використовуються для створення пунктів меню, об'єктів класу MenuItem, та для задавання загальних властивостей для всього меню. Невізуальні об'єкти не розміщаються безпосередньо на формі, а розташовані в окремій області під нею у вікні Конструктора (рис. 3.7).

Оскільки пункти контекстного меню є елементами того ж класу, що і пункти головного меню, то створення обробника події для контекстного меню відбувається так само, як і було описано для обробників подій головного меню.

На відміну від головного меню, яке може мати лише форма, різні контекстні меню можуть мати різні візуальні об'єкти, наприклад такі як panel, groupBox, button та інші. Відповідно, щоб зв'язати певне контекстне меню з певним об'єктом, у кожного об'єкту, якій може мати контекстне меню, є властивість ContextMenuStrip, в якій вказується контекстне меню, що відображатиметься при натисканні правої кнопки миші на цьому об'єкті.

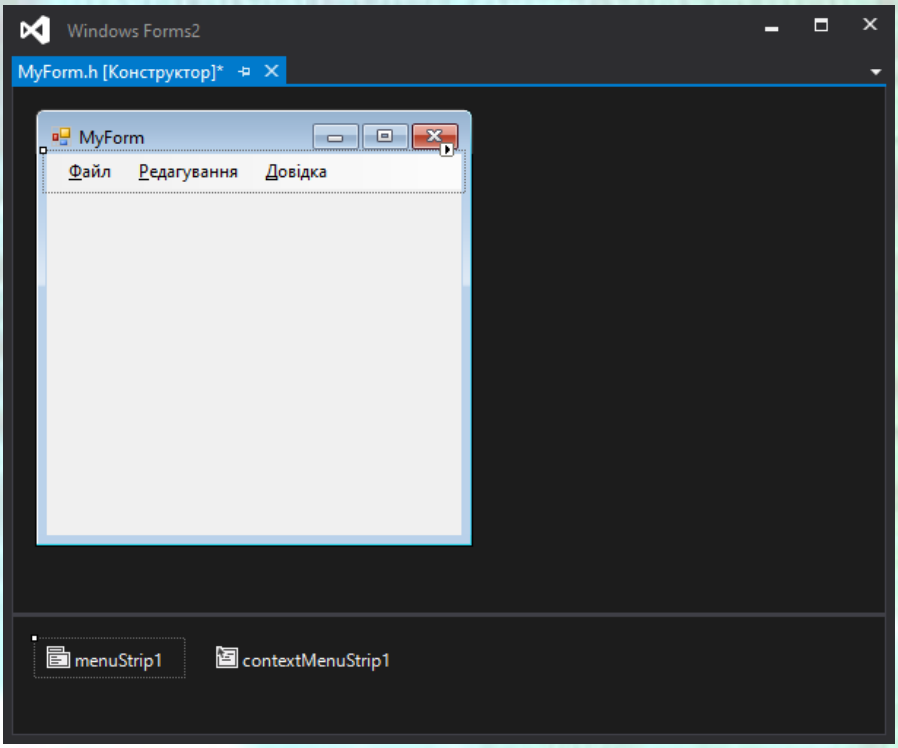

Рисунок 3.7 – Вікно Конструктора з невізуальними об'єктами `menuStrip1` та `contextMenuStrip1`

3.4 Додавання панелі інструментів

Для створення у вікні програми панелі інструментів, необхідно у вікні Панель елементів (Toolbox) з розділу Меню і панелі інструментів (Menus & Toolbars) обрати елемент ToolStrip та розмістити його у вікні редактора форми (рис. 3.8).

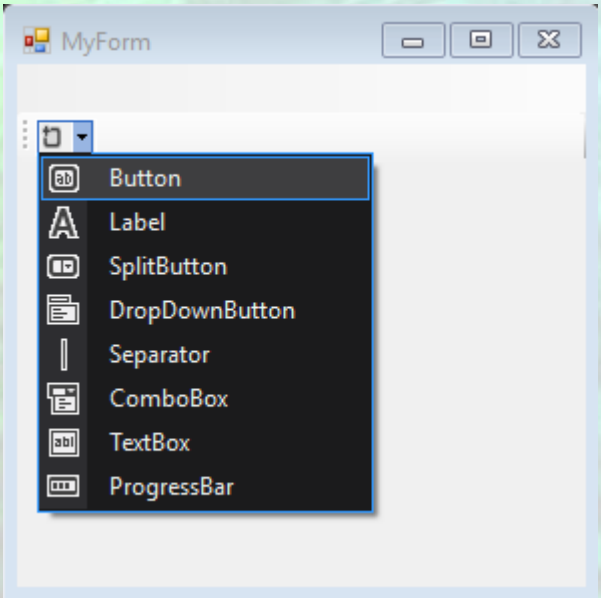

Рисунок 3.8 – Створення панелі інструментів у вікні редактора форми

Як видно з рис. 3.8, на панелі інструментів можна розмістити об'єкти класів, наведених в табл. 3.2.

Таблиця 3.2 – Об'єкти панелі інструментів

Ім'я класу	Опис
ToolStripButton	Являє собою обираний елемент ToolStripItem та може містити текст і зображення.
ToolStripLabel	Представляє елемент ToolStripItem, який не виділяється, відображає текст і зображення та може відображати гіперпосилання.
ToolStripSplitButton	Являє поєднання стандартної кнопки зліва і кнопки з переліком праворуч або навпаки, якщо значення властивості RightToLeft є Yes.
ToolStripDropDownButton	Представляє елемент керування, який при натисканні показує пов'язаний елемент ToolStripDropDown, з якого користувач може вибрати один елемент.
ToolStripSeparator	Являє собою роздільник-лінію, який використовуються для групування елементів з ToolStrip або елементів списку MenuStrip або ContextMenuStrip або інших ToolStripDropDown елементів управління.
ToolStripComboBox	Являє собою елемент ComboBox, який правильно промальовується в ToolStrip.
ToolStripTextBox	Являє собою текстове поле в ToolStrip, що дозволяє користувачеві вводити текст.
ToolStripProgressBar	Являє собою панель управління ходу виконання Windows, що міститься в StatusStrip.

Для додавання реакції на подію до елементу панелі інструментів необхідно двічі клацнути мишею на потрібному елементі панелі у вікні Конструктора (Design) форми. Можна також виділити елемент панелі інструментів у вікні Конструктора та перейти до вікна Властивості (Properties). У вікні Властивості необхідно натиснути кнопку Events (Події) для відкриття відповідної вкладки. На вкладці Events вікна Властивості потрібно в розділі Action обрати подію Click і двічі клацнути у пустому полі праворуч (рис. 3.4).

Результатом цих дій буде створення пустої функції – обробника події Click, тобто реакції на клацання мишею по даному елементу панелі. При цьому відкриється вікно редактору коду з курсором всередині тіла цієї функції.

```
private: System::Void toolStripComboBox1_Click(System::Object^  sender, System::EventArgs^
e) {
        }
```

Тепер після відкритої фігурної дужки можна писати код реакції на дану подію.

У вікні Властивості на вкладці Events (Події) відображаються можливі події для виділенного у даний момент об’єкту у вікні редактора форми. Щоб оглянути події іншого об’єкту, його можна обрату у спадаючому списку у верхній частині вікна Властивості (Properties).

Приклад програмної реалізації

Приклад створення проекту Windows Forms з використанням класів MenuStrip, ContextMenuStrip та ToolStrip.

Необхідно створити додаток, який розраховуватиме витрату потужності на перемішування для пропелерної мішалки. У якості вихідних даних повинні задаватись змінні за допомогою текстових полів.

Для введення вхідних даних додаємо до форми 4 об'єкти класу TextBox та 4 об'єкти класу Label для підпису полів введення. Даємо імена доданим об'єктам в залежності від змінних, які будуть вводитись: label_Kn; textBox_Kn; label_n; textBox_n; label_Rc; textBox_Rc; label_dm; textBox_dm.

У властивості Text об'єктів label записуємо пояснення до кожного поля введення. У файлі MyForm.h внаслідок цього додадуться наступні рядки коду у відповідних розділах:

```
this->label_Kn->Text = L"Коефіцієнт Kn";
this->label_n->Text = L"Швидкість мішалки, об/с";
this->label_Rc->Text = L"Густина рідини, кг/м3";
this->label_dm->Text = L"Діаметр мішалки, м";
```

Надалі додаємо елемент головного меню MenuStrip до форми. За допомогою конструктора меню додаємо чотрири пункти головного меню та один роздільник (у властивості Text для цього об'єкту ставиться символ «-») - об'єкти класу ToolStripMenuItem. Назви пунктів меню зберігаються у властивості Text об'єктів ToolStripMenuItem. Щоб додати до пунктів меню сполучення швидкіх клавіш для можливості виклику події обробки вибору певного пункту меню, необхідно скористатись властивістю ShortcutKeys об'єктів ToolStripMenuItem. Для додавання піктограм підказок до пунктів меню слід у властивості Image обрати потрібний файл графічного зображення.

Щоб додати обробник подій для пункту меню необхідно двічі клацнути мишею по ньому у вікні редактора форми або обрати потрібний пункт меню та на вкладці Events (Події) вікна Властивості (Properties) та в розділі Action обрати подію Click і двічі клацнути у пустому полі праворуч (рис. 3.4). Як наслідок, з'явиться пуста функція обробника події Click для відповідного пункту меню, де в подальшому потрібно буде додати програмний код.

Після виконання всіх цих дій, у файлі MyForm.h додасться автоматично наступний код:

```
//
// menuStrip1
//
this->menuStrip1->Items->AddRange(gcnew cli::array<
        System::Windows::Forms::ToolStripItem^  >(1) { this->розрахунокToolStripMenuItem });
this->menuStrip1->Location = System::Drawing::Point(0, 0);
```

```
this->menuStrip1->Name = L"menuStrip1";
this->menuStrip1->Size = System::Drawing::Size(491, 24);
this->menuStrip1->TabIndex = 0;
this->menuStrip1->Text = L"menuStrip1";
//
// розрахунокToolStripMenuItem
//
this->розрахунокToolStripMenuItem->DropDownItems->
       AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^  >(5) {
       this->тестToolStripMenuItem,
       this->очиститиToolStripMenuItem,
       this->розраховатиToolStripMenuItem,
       this->toolStripMenuItem1,
       this->вихідToolStripMenuItem
       });
this->розрахунокToolStripMenuItem->Name = L"розрахунокToolStripMenuItem";
this->розрахунокToolStripMenuItem->Size = System::Drawing::Size(82, 20);
this->розрахунокToolStripMenuItem->Text = L"Розрахунок";
//
// тестToolStripMenuItem
//
this->тестToolStripMenuItem->Image = (cli::safe_cast<System::Drawing::Image^>
       (resources->GetObject(L"тестToolStripMenuItem.Image")));
this->тестToolStripMenuItem->Name = L"тестToolStripMenuItem";
this->тестToolStripMenuItem->ShortcutKeys =
       static_cast<System::Windows::Forms::Keys>((System::Windows::Forms::Keys::Control |
       System::Windows::Forms::Keys::T));
this->тестToolStripMenuItem->Size = System::Drawing::Size(206, 22);
this->тестToolStripMenuItem->Text = L"&Тест";
this->тестToolStripMenuItem->Click += gcnew System::EventHandler
       (this, &MyForm::тестToolStripMenuItem_Click);
//
// очиститиToolStripMenuItem
//
this->очиститиToolStripMenuItem->Image = (cli::safe_cast<System::Drawing::Image^>
       (resources->GetObject(L"очиститиToolStripMenuItem.Image")));
this->очиститиToolStripMenuItem->Name = L"очиститиToolStripMenuItem";
this->очиститиToolStripMenuItem->ShortcutKeys =
       static_cast<System::Windows::Forms::Keys>((System::Windows::Forms::Keys::Alt |
       System::Windows::Forms::Keys::O));
this->очиститиToolStripMenuItem->Size = System::Drawing::Size(206, 22);
this->очиститиToolStripMenuItem->Text = L"&Очистити";
this->очиститиToolStripMenuItem->Click += gcnew System::EventHandler
       (this, &MyForm::очиститиToolStripMenuItem_Click);
//
// розраховатиToolStripMenuItem
//
this->розраховатиToolStripMenuItem->Image =
       (cli::safe_cast<System::Drawing::Image^>
       (resources->GetObject(L"розраховатиToolStripMenuItem.Image")));
this->розраховатиToolStripMenuItem->Name = L"розраховатиToolStripMenuItem";
this->розраховатиToolStripMenuItem->ShortcutKeys = static_cast<System::Windows::Forms::Keys>
       (((System::Windows::Forms::Keys::Control | System::Windows::Forms::Keys::Alt) |
       System::Windows::Forms::Keys::C));
this->розраховатиToolStripMenuItem->Size = System::Drawing::Size(206, 22);
this->розраховатиToolStripMenuItem->Text = L"&Розрахувати";
this->розраховатиToolStripMenuItem->Click +=
       gcnew System::EventHandler(this, &MyForm::розраховатиToolStripMenuItem_Click);
//
// toolStripMenuItem1
//
```

```
this->toolStripMenuItem1->Name = L"toolStripMenuItem1";
this->toolStripMenuItem1->Size = System::Drawing::Size(203, 6);
//
// вихідToolStripMenuItem
//
this->вихідToolStripMenuItem->Image = (cli::safe_cast<System::Drawing::Image^>
        (resources->GetObject(L"вихідToolStripMenuItem.Image")));
this->вихідToolStripMenuItem->Name = L"вихідToolStripMenuItem";
this->вихідToolStripMenuItem->ShortcutKeys = static_cast<System::Windows::Forms::Keys>
        ((System::Windows::Forms::Keys::Alt | System::Windows::Forms::Keys::F4));
this->вихідToolStripMenuItem->Size = System::Drawing::Size(206, 22);
this->вихідToolStripMenuItem->Text = L"Ви&хід";
this->вихідToolStripMenuItem->Click += gcnew System::EventHandler
        (this, &MyForm::вихідToolStripMenuItem_Click);
```

Внаслідок виконання вище зазначених дій, редактор форми матиме наступний вигляд (рис. 3.9).

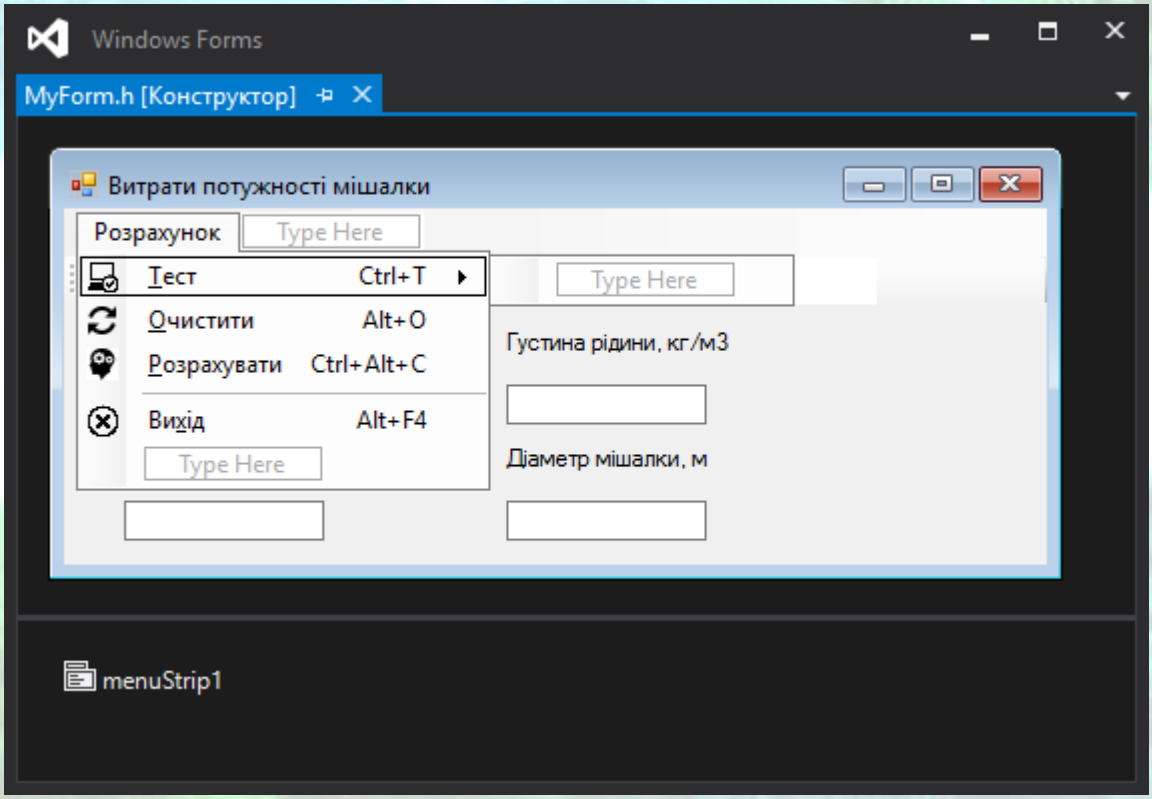

Рисунок 3.9 – Створення головного меню у вікні редактора форми

Для додавання контекстного меню необхідно розмістити у вікні редактора форми об'єкт класу ContextMenuStrip. Щоб контекстне меню відображалося після натискання правої кнопки миші у межах вікна програми, необхідно у властивості форми ContextMenuStrip обрати відповідний об'єкт контекстного меню.

Створення пунктів контекстного меню повністю аналогічне створенню пунктів головного меню. Пункти контекстного меню зазвичай можуть повторювати дії, що виконуються в головному меню. В цьому випадку для

створення обробника події не потрібно заново писати той самий код. Для пункту контекстного меню треба перейти до вкладки Events (Події) вікна Властивості (Properties) та в розділі Action обрати подію Click. У пустому полі праворуч вже буде випадаючий перелік існуючих функцій обробки подій головного меню. Із цього переліку потрібно обрати відповідну функцію, що була створена для обробки події аналогічного пункту головного меню (рис. 3.10).

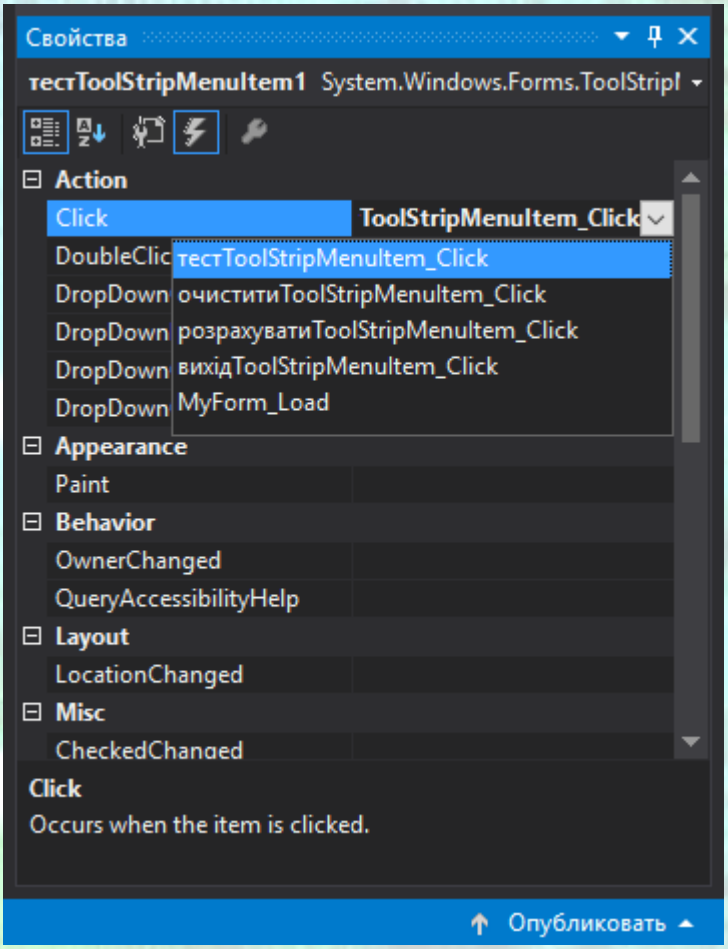

Рисунок 3.10 – Вибір обробника події для пункту меню

По завершенню створення пунктів контекстного меню, вікно редактора форми повинне мати вигляд, як зображено на рис. 3.11.

Після виконання всіх необхідних дій по створенню контекстного меню, до файлу MyForm.h додастся автоматично створений програмний код.

```
//
// contextMenuStrip1
//
this->contextMenuStrip1->Items->AddRange(gcnew cli::array<
	System::Windows::Forms::ToolStripItem^  >(3) {
		this->тестToolStripMenuItem1,
		this->очиститиToolStripMenuItem1, this->розрахуватиToolStripMenuItem1
	});
this->contextMenuStrip1->Name = L"contextMenuStrip1";
```

```
this->contextMenuStrip1->Size = System::Drawing::Size(207, 70);
//
// тестToolStripMenuItem1
//
this->тестToolStripMenuItem1->Image = (cli::safe_cast<System::Drawing::Image^>
       (resources->GetObject(L"тестToolStripMenuItem1.Image")));
this->тестToolStripMenuItem1->Name = L"тестToolStripMenuItem1";
this->тестToolStripMenuItem1->ShortcutKeys =
       static_cast<System::Windows::Forms::Keys>((System::Windows::Forms::Keys::Control |
       System::Windows::Forms::Keys::T));
this->тестToolStripMenuItem1->Size = System::Drawing::Size(206, 22);
this->тестToolStripMenuItem1->Text = L"&Тест";
this->тестToolStripMenuItem1->Click += gcnew System::EventHandler(this,
       &MyForm::тестToolStripMenuItem_Click);
//
// очиститиToolStripMenuItem1
//
this->очиститиToolStripMenuItem1->Image =
       (cli::safe_cast<System::Drawing::Image^>(resources-
       >GetObject(L"очиститиToolStripMenuItem1.Image")));
this->очиститиToolStripMenuItem1->Name = L"очиститиToolStripMenuItem1";
this->очиститиToolStripMenuItem1->ShortcutKeys =
       static_cast<System::Windows::Forms::Keys>((System::Windows::Forms::Keys::Alt |
       System::Windows::Forms::Keys::O));
this->очиститиToolStripMenuItem1->Size = System::Drawing::Size(206, 22);
this->очиститиToolStripMenuItem1->Text = L"&Очистити";
this->очиститиToolStripMenuItem1->Click +=
       gcnew System::EventHandler(this, &MyForm::очиститиToolStripMenuItem_Click);
//
// розрахуватиToolStripMenuItem1
//
this->розрахуватиToolStripMenuItem1->Image =
       (cli::safe_cast<System::Drawing::Image^>(resources->
       GetObject(L"розрахуватиToolStripMenuItem1.Image")));
this->розрахуватиToolStripMenuItem1->Name = L"розрахуватиToolStripMenuItem1";
this->розрахуватиToolStripMenuItem1->ShortcutKeys =
       static_cast<System::Windows::Forms::Keys>(((System::Windows::Forms::Keys::Control |
       System::Windows::Forms::Keys::Alt) | System::Windows::Forms::Keys::C));
this->розрахуватиToolStripMenuItem1->Size = System::Drawing::Size(206, 22);
this->розрахуватиToolStripMenuItem1->Text = L"&Розрахувати";
this->розрахуватиToolStripMenuItem1->Click +=
       gcnew System::EventHandler(this, &MyForm::розрахуватиToolStripMenuItem_Click);
```

Надалі нам необхідно додати панель інструментів до вікна нашої програми. Для цього розміщуємо у вікні редактора форми об'єкт класу `ToolStrip`. Уредакторі панелі інструментів додаємо чотири кнопки – об'єкти класу `ToolStripButton`, один роздільник - об'єкт класу `ToolStripSeparator`, одне текстове поле – об'єкт класу `ToolStripLabel` та одне поле введення - об'єкт класу `ToolStripTextBox`. Чотири кнопки панелі інструментів будуть відповідати чотирьом пунктам головного меню, які буде відділяти роздільник від поля `Label` та поля `TextBox`. Об'єктам панелі інструментів надаємо імена в полі Name в залежності від їх призначення.

Для того, щоб на кнопках `ToolStripButton` панелі інструментів відображався окрім малюнка ще й текст, значення властивості `DisplayStyle`

для чотирьох кнопок панелі інструментів у вікні Властивості (Properties) міняємо на ImageAndText. Після цього у властивості Text вказуємо текст, а у властивості Image обираємо малюнок, які будуть відображатися на кнопці.

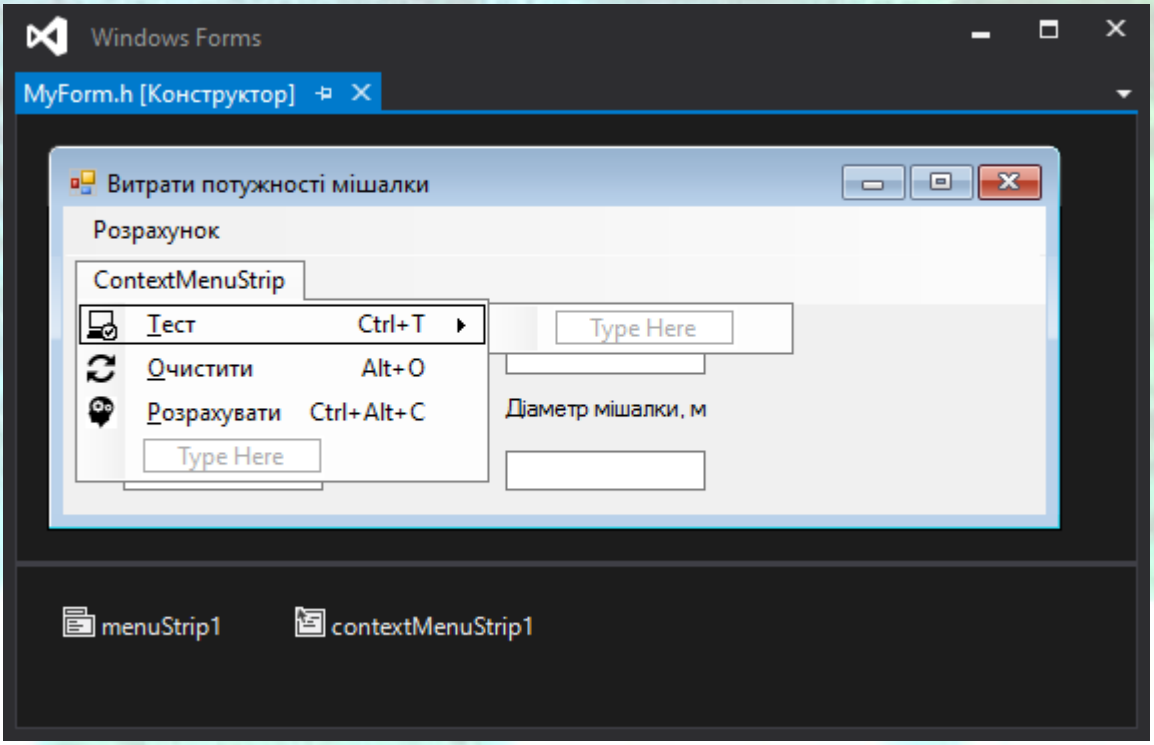

Рисунок 3.11 – Створення контекстного меню у вікні редактора форми

Обробник натискання на кнопку панелі інструментів - подія Click розташована у вкладці Events (Події) вікна Властивості (Properties) в розділі Action. Оскільки обробники подій у нас вже створені для пунктів головного меню, у пустому полі праворуч події Click з випадаючого переліку існуючих функцій обробки подій обираємо потрібну. Дану операцію повторюємо для всіх чотирьох кнопок панелі інструментів.

По завершенню створення панелі інструментів до файлу MyForm.h автоматично буде додано наступний програмний код.

```
//
// toolStrip1
//
this->toolStrip1->Items->AddRange(gcnew cli::array<
        System::Windows::Forms::ToolStripItem^  >(7) {
        this->toolStripButton_Тест,
        this->toolStripButton_Розрахунок,
        this->toolStripButton_Очищення,
        this->toolStripButton_Вихід,
        this->toolStripSeparator1,
        this->toolStripLabel_N,
        this->toolStripTextBox_N
        });
this->toolStrip1->Location = System::Drawing::Point(0, 24);
```

```
this->toolStrip1->Name = L"toolStrip1";
this->toolStrip1->Size = System::Drawing::Size(491, 25);
this->toolStrip1->TabIndex = 2;
this->toolStrip1->Text = L"toolStrip1";
//
// toolStripButton_Тест
//
this->toolStripButton_Тест->Image = (cli::safe_cast<System::Drawing::Image^>
       (resources->GetObject(L"toolStripButton_Тест.Image")));
this->toolStripButton_Тест->ImageTransparentColor = System::Drawing::Color::Magenta;
this->toolStripButton_Тест->Name = L"toolStripButton_Тест";
this->toolStripButton_Тест->Size = System::Drawing::Size(51, 22);
this->toolStripButton_Тест->Text = L"Тест";
this->toolStripButton_Тест->Click += gcnew System::EventHandler(this,
       &MyForm::тестToolStripMenuItem_Click);
//
// toolStripButton_Розрахунок
//
this->toolStripButton_Розрахунок->Image = (cli::safe_cast<System::Drawing::Image^>
       (resources->GetObject(L"toolStripButton_Розрахунок.Image")));
this->toolStripButton_Розрахунок->ImageTransparentColor = System::Drawing::Color::Magenta;
this->toolStripButton_Розрахунок->Name = L"toolStripButton_Розрахунок";
this->toolStripButton_Розрахунок->Size = System::Drawing::Size(94, 22);
this->toolStripButton_Розрахунок->Text = L"Розрахувати";
this->toolStripButton_Розрахунок->Click +=
       gcnew System::EventHandler(this, &MyForm::розрахуватиToolStripMenuItem_Click);
//
// toolStripButton_Очищення
//
this->toolStripButton_Очищення->Image = (cli::safe_cast<System::Drawing::
       Image^>(resources->GetObject(L"toolStripButton_Очищення.Image")));
this->toolStripButton_Очищення->ImageTransparentColor = System::Drawing::Color::Magenta;
this->toolStripButton_Очищення->Name = L"toolStripButton_Очищення";
this->toolStripButton_Очищення->Size = System::Drawing::Size(80, 22);
this->toolStripButton_Очищення->Text = L"Очистити";
this->toolStripButton_Очищення->Click +=
       gcnew System::EventHandler(this, &MyForm::очиститиToolStripMenuItem_Click);
//
// toolStripButton_Вихід
//
this->toolStripButton_Вихід->Image = (cli::safe_cast<System::Drawing::
       Image^>(resources->GetObject(L"toolStripButton_Вихід.Image")));
this->toolStripButton_Вихід->ImageTransparentColor = System::Drawing::Color::Magenta;
this->toolStripButton_Вихід->Name = L"toolStripButton_Вихід";
this->toolStripButton_Вихід->Size = System::Drawing::Size(55, 22);
this->toolStripButton_Вихід->Text = L"Вихід";
this->toolStripButton_Вихід->Click += gcnew System::EventHandler(this,
       &MyForm::вихідToolStripMenuItem_Click);
//
// toolStripSeparator1
//
this->toolStripSeparator1->Name = L"toolStripSeparator1";
this->toolStripSeparator1->Size = System::Drawing::Size(6, 25);
//
// toolStripLabel_N
//
this->toolStripLabel_N->Name = L"toolStripLabel_N";
this->toolStripLabel_N->Size = System::Drawing::Size(10, 22);
this->toolStripLabel_N->Text = L" ";
//
// toolStripTextBox_N
```

```
//
this->toolStripTextBox_N->Name = L"toolStripTextBox_N";
this->toolStripTextBox_N->Size = System::Drawing::Size(100, 25);
```

Після запуску вікно нашої програми з головним та контекстним меню та панеллю інструментів матиме вигляд, як зображено на рис. 3.12.

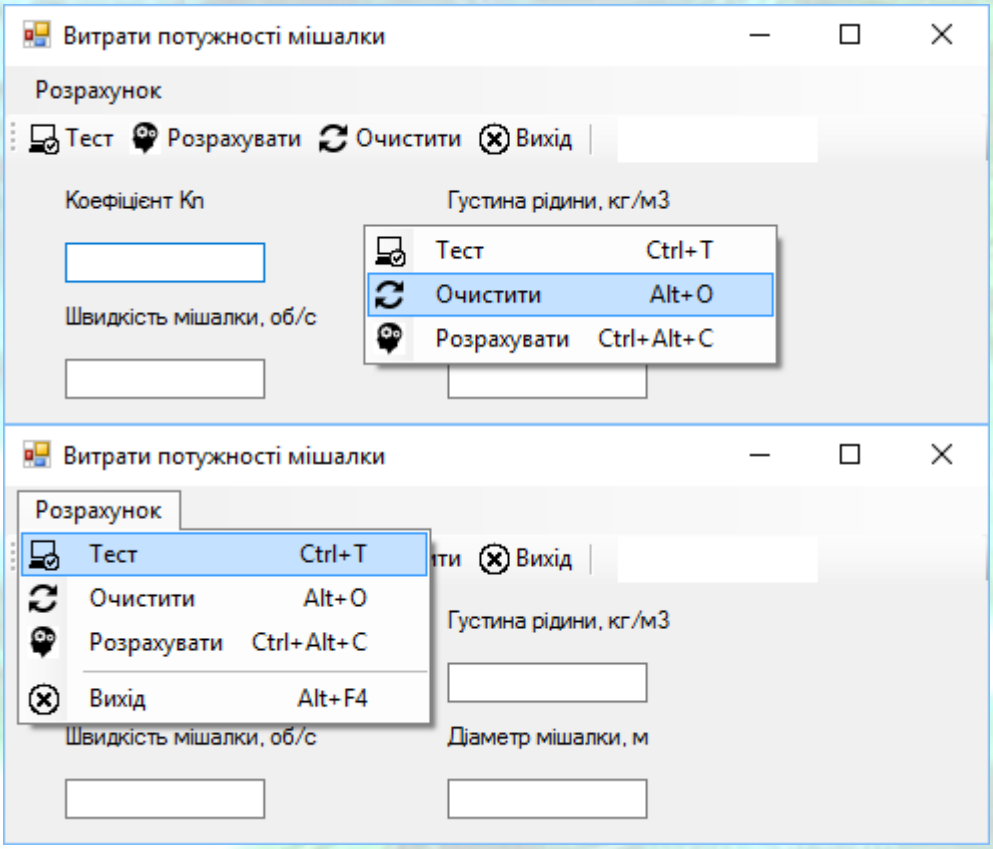

Рисунок 3.12 – Вікно програми з панеллю інструментів та меню

Для того, щоб обробники подій головного меню запрацювали, необхідно додати в них відповідний програмний код, як показано нижче.

```
#pragma endregion

private: System::Void тестToolStripMenuItem_Click(System::Object^  sender,
System::EventArgs^  e) {
       //Задавання значень змінних для тестового розрахунку
       textBox_Kn->Text = "0,32";
       textBox_n->Text = "3,96";
       textBox_Rc->Text = "1200";
       textBox_dm->Text = "2,8";
}
private: System::Void очиститиToolStripMenuItem_Click(System::Object^  sender,
System::EventArgs^  e) {
       //Очищення поля введення
       textBox_Kn->Text = "";
       textBox_n->Text = "";
       textBox_Rc->Text = "";
       textBox_dm->Text = "";
```

```
        toolStripLabel_N->Text = "";
        toolStripTextBox_N->Text = "";
}
private: System::Void розрахуватиToolStripMenuItem_Click(System::Object^  sender,
System::EventArgs^  e) {
        float Kn, n, Rc, dm, N;
        //Перетворює рядкове представлення числа в дійсне
        Kn = float::Parse(textBox_Kn->Text);
        n = float::Parse(textBox_n->Text);
        Rc = float::Parse(textBox_Rc->Text);
        dm = float::Parse(textBox_dm->Text);
        N = Kn*pow(n, 3)*Rc*pow(dm, 5);
        toolStripLabel_N->Text = "Витрати потужності на перемішування ";
        MessageBox::Show("Витрати потужності на перемішування = " + N.ToString("F"));
        toolStripTextBox_N->Text = N.ToString("F");
}
private: System::Void вихідToolStripMenuItem_Click(System::Object^  sender,
System::EventArgs^  e) {
        Close(); //Закрити програму
}
```

Як видно з програмного коду обробки події `розрахуватиToolStripMenuItem_Click`, розраховане значення втрати потужності мішалки ми виводимо в діалоговому вікні `MessageBox`, а також у полі `toolStripTextBox_N` з поясненням в `toolStripLabel_N` на панелі інструментів. Результат роботи програми наведений на рис. 3.13.

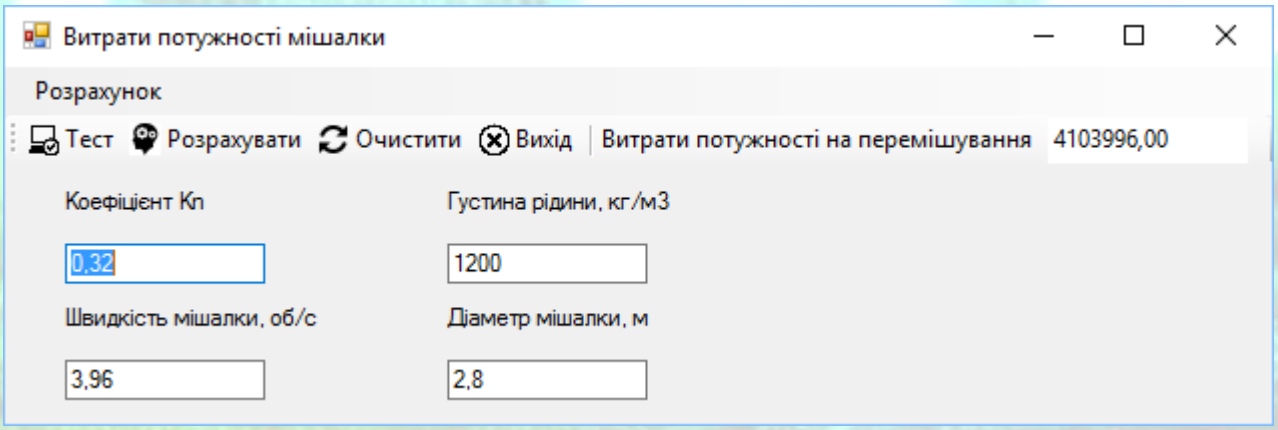

Рисунок 3.13 – Вікно програми з результатами розрахунку

Повний код створеного програмного модуля наведений нижче в лістингу 3.1 – файл `MyForm.cpp`, та лістингу 3.2 – заголовочний файл форми `MyForm.h`.

Програмний код

Лістинг 3.1

```
//Програмний код модуля MyForm.cpp
#include "MyForm.h"

using namespace WindowsForms; //ім'я вашого проекту

[STAThreadAttribute]
int main(array<System::String ^> ^args)
{
	Application::EnableVisualStyles();
	Application::SetCompatibleTextRenderingDefault(false);
	Application::Run(gcnew MyForm());
	return 0;
}
```

Лістинг 3.2

```
//Програмний код модуля MyForm.h
#pragma once

namespace WindowsForms {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;
	#include <math.h> //Підключення бібліотеки математичних функцій

	/// <summary>
	/// Сводка для MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
		}

	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::MenuStrip^  menuStrip1;
	private: System::Windows::Forms::ContextMenuStrip^  contextMenuStrip1;
	private: System::Windows::Forms::ToolStrip^  toolStrip1;
	private: System::Windows::Forms::ToolStripButton^  toolStripButton_Тест;
	private: System::Windows::Forms::ToolStripSeparator^  toolStripSeparator1;
	private: System::Windows::Forms::ToolStripMenuItem^  розрахунокToolStripMenuItem;
	private: System::Windows::Forms::ToolStripMenuItem^  тестToolStripMenuItem;
	private: System::Windows::Forms::ToolStripMenuItem^  очиститиToolStripMenuItem;
	private: System::Windows::Forms::ToolStripMenuItem^  розрахуватиToolStripMenuItem;
	private: System::Windows::Forms::ToolStripSeparator^  toolStripMenuItem1;
	private: System::Windows::Forms::ToolStripMenuItem^  вихідToolStripMenuItem;
	private: System::Windows::Forms::ToolStripButton^  toolStripButton_Розрахунок;
	private: System::Windows::Forms::ToolStripButton^  toolStripButton_Очищення;
	private: System::Windows::Forms::ToolStripButton^  toolStripButton_Вихід;
	private: System::Windows::Forms::ToolStripTextBox^  toolStripTextBox_N;
	private: System::Windows::Forms::Label^  label_Kn;
	private: System::Windows::Forms::TextBox^  textBox_Kn;
	private: System::Windows::Forms::TextBox^  textBox_n;
	private: System::Windows::Forms::Label^  label_n;
	private: System::Windows::Forms::TextBox^  textBox_Rc;
```

```
        private: System::Windows::Forms::Label^  label_Rc;
        private: System::Windows::Forms::TextBox^  textBox_dm;
        private: System::Windows::Forms::Label^  label_dm;
        private: System::Windows::Forms::ToolStripLabel^  toolStripLabel_N;
        private: System::Windows::Forms::ToolStripMenuItem^  тестToolStripMenuItem1;
        private: System::Windows::Forms::ToolStripMenuItem^  очиститиToolStripMenuItem1;
        private: System::Windows::Forms::ToolStripMenuItem^  розрахуватиToolStripMenuItem1;
        private: System::ComponentModel::IContainer^  components;

        protected:

        private:
                /// <summary>
                /// Обязательная переменная конструктора.
                /// </summary>


#pragma region Windows Form Designer generated code
                /// <summary>
                /// Требуемый метод для поддержки конструктора — не изменяйте
                /// содержимое этого метода с помощью редактора кода.
                /// </summary>
                void InitializeComponent(void)
                {
                        this->components = (gcnew System::ComponentModel::Container());
                        System::ComponentModel::ComponentResourceManager^  resources = (gcnew
System::ComponentModel::ComponentResourceManager(MyForm::typeid));
                        this->menuStrip1 = (gcnew System::Windows::Forms::MenuStrip());
                        this->розрахунокToolStripMenuItem = (gcnew
System::Windows::Forms::ToolStripMenuItem());
                        this->тестToolStripMenuItem = (gcnew
System::Windows::Forms::ToolStripMenuItem());
                        this->очиститиToolStripMenuItem = (gcnew
System::Windows::Forms::ToolStripMenuItem());
                        this->розрахуватиToolStripMenuItem = (gcnew
System::Windows::Forms::ToolStripMenuItem());
                        this->toolStripMenuItem1 = (gcnew
System::Windows::Forms::ToolStripSeparator());
                        this->вихідToolStripMenuItem = (gcnew
System::Windows::Forms::ToolStripMenuItem());
                        this->contextMenuStrip1 = (gcnew
System::Windows::Forms::ContextMenuStrip(this->components));
                        this->тестToolStripMenuItem1 = (gcnew
System::Windows::Forms::ToolStripMenuItem());
                        this->очиститиToolStripMenuItem1 = (gcnew
System::Windows::Forms::ToolStripMenuItem());
                        this->розрахуватиToolStripMenuItem1 = (gcnew
System::Windows::Forms::ToolStripMenuItem());
                        this->toolStrip1 = (gcnew System::Windows::Forms::ToolStrip());
                        this->toolStripButton_Тест = (gcnew
System::Windows::Forms::ToolStripButton());
                        this->toolStripButton_Розрахунок = (gcnew
System::Windows::Forms::ToolStripButton());
                        this->toolStripButton_Очищення = (gcnew
System::Windows::Forms::ToolStripButton());
                        this->toolStripButton_Вихід = (gcnew
System::Windows::Forms::ToolStripButton());
                        this->toolStripSeparator1 = (gcnew
System::Windows::Forms::ToolStripSeparator());
                        this->toolStripLabel_N = (gcnew
System::Windows::Forms::ToolStripLabel());
```

```
                    this->toolStripTextBox_N = (gcnew
System::Windows::Forms::ToolStripTextBox());
                    this->label_Kn = (gcnew System::Windows::Forms::Label());
                    this->textBox_Kn = (gcnew System::Windows::Forms::TextBox());
                    this->textBox_n = (gcnew System::Windows::Forms::TextBox());
                    this->label_n = (gcnew System::Windows::Forms::Label());
                    this->textBox_Rc = (gcnew System::Windows::Forms::TextBox());
                    this->label_Rc = (gcnew System::Windows::Forms::Label());
                    this->textBox_dm = (gcnew System::Windows::Forms::TextBox());
                    this->label_dm = (gcnew System::Windows::Forms::Label());
                    this->menuStrip1->SuspendLayout();
                    this->contextMenuStrip1->SuspendLayout();
                    this->toolStrip1->SuspendLayout();
                    this->SuspendLayout();
                    //
                    // menuStrip1
                    //
                    this->menuStrip1->Items->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^  >(1) { this->розрахунокToolStripMenuItem });
                    this->menuStrip1->Location = System::Drawing::Point(0, 0);
                    this->menuStrip1->Name = L"menuStrip1";
                    this->menuStrip1->Size = System::Drawing::Size(491, 24);
                    this->menuStrip1->TabIndex = 0;
                    this->menuStrip1->Text = L"menuStrip1";
                    //
                    // розрахунокToolStripMenuItem
                    //
                    this->розрахунокToolStripMenuItem->DropDownItems->AddRange(gcnew
cli::array< System::Windows::Forms::ToolStripItem^  >(5) {
                            this->тестToolStripMenuItem,
                                    this->очиститиToolStripMenuItem, this-
>розрахуватиToolStripMenuItem, this->toolStripMenuItem1, this->вихідToolStripMenuItem
                    });
                    this->розрахунокToolStripMenuItem->Name =
L"розрахунокToolStripMenuItem";
                    this->розрахунокToolStripMenuItem->Size = System::Drawing::Size(82,
20);
                    this->розрахунокToolStripMenuItem->Text = L"Розрахунок";
                    //
                    // тестToolStripMenuItem
                    //
                    this->тестToolStripMenuItem->Image =
(cli::safe_cast<System::Drawing::Image^>(resources-
>GetObject(L"тестToolStripMenuItem.Image")));
                    this->тестToolStripMenuItem->Name = L"тестToolStripMenuItem";
                    this->тестToolStripMenuItem->ShortcutKeys =
static_cast<System::Windows::Forms::Keys>((System::Windows::Forms::Keys::Control |
System::Windows::Forms::Keys::T));
                    this->тестToolStripMenuItem->Size = System::Drawing::Size(206, 22);
                    this->тестToolStripMenuItem->Text = L"&Тест";
                    this->тестToolStripMenuItem->Click += gcnew System::EventHandler(this,
&MyForm::тестToolStripMenuItem_Click);
                    //
                    // очиститиToolStripMenuItem
                    //
                    this->очиститиToolStripMenuItem->Image =
(cli::safe_cast<System::Drawing::Image^>(resources-
>GetObject(L"очиститиToolStripMenuItem.Image")));
                    this->очиститиToolStripMenuItem->Name = L"очиститиToolStripMenuItem";
```

```
                    this->очиститиToolStripMenuItem->ShortcutKeys =
static_cast<System::Windows::Forms::Keys>((System::Windows::Forms::Keys::Alt |
System::Windows::Forms::Keys::O));
                    this->очиститиToolStripMenuItem->Size = System::Drawing::Size(206, 22);
                    this->очиститиToolStripMenuItem->Text = L"&Очистити";
                    this->очиститиToolStripMenuItem->Click += gcnew
System::EventHandler(this, &MyForm::очиститиToolStripMenuItem_Click);
                    //
                    // розрахуватиToolStripMenuItem
                    //
                    this->розрахуватиToolStripMenuItem->Image =
(cli::safe_cast<System::Drawing::Image^>(resources-
>GetObject(L"розрахуватиToolStripMenuItem.Image")));
                    this->розрахуватиToolStripMenuItem->Name =
L"розрахуватиToolStripMenuItem";
                    this->розрахуватиToolStripMenuItem->ShortcutKeys =
static_cast<System::Windows::Forms::Keys>(((System::Windows::Forms::Keys::Control |
System::Windows::Forms::Keys::Alt)
                            | System::Windows::Forms::Keys::C));
                    this->розрахуватиToolStripMenuItem->Size = System::Drawing::Size(206,
22);
                    this->розрахуватиToolStripMenuItem->Text = L"&Розрахувати";
                    this->розрахуватиToolStripMenuItem->Click += gcnew
System::EventHandler(this, &MyForm::розрахуватиToolStripMenuItem_Click);
                    //
                    // toolStripMenuItem1
                    //
                    this->toolStripMenuItem1->Name = L"toolStripMenuItem1";
                    this->toolStripMenuItem1->Size = System::Drawing::Size(203, 6);
                    //
                    // вихідToolStripMenuItem
                    //
                    this->вихідToolStripMenuItem->Image =
(cli::safe_cast<System::Drawing::Image^>(resources-
>GetObject(L"вихідToolStripMenuItem.Image")));
                    this->вихідToolStripMenuItem->Name = L"вихідToolStripMenuItem";
                    this->вихідToolStripMenuItem->ShortcutKeys =
static_cast<System::Windows::Forms::Keys>((System::Windows::Forms::Keys::Alt |
System::Windows::Forms::Keys::F4));
                    this->вихідToolStripMenuItem->Size = System::Drawing::Size(206, 22);
                    this->вихідToolStripMenuItem->Text = L"Ви&хід";
                    this->вихідToolStripMenuItem->Click += gcnew System::EventHandler(this,
&MyForm::вихідToolStripMenuItem_Click);
                    //
                    // contextMenuStrip1
                    //
                    this->contextMenuStrip1->Items->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^  >(3) {
                            this->тестToolStripMenuItem1,
                                    this->очиститиToolStripMenuItem1, this-
>розрахуватиToolStripMenuItem1
                    });
                    this->contextMenuStrip1->Name = L"contextMenuStrip1";
                    this->contextMenuStrip1->Size = System::Drawing::Size(207, 70);
                    //
                    // тестToolStripMenuItem1
                    //
                    this->тестToolStripMenuItem1->Image =
(cli::safe_cast<System::Drawing::Image^>(resources-
>GetObject(L"тестToolStripMenuItem1.Image")));
                    this->тестToolStripMenuItem1->Name = L"тестToolStripMenuItem1";
```

```
                    this->тестToolStripMenuItem1->ShortcutKeys =
static_cast<System::Windows::Forms::Keys>((System::Windows::Forms::Keys::Control |
System::Windows::Forms::Keys::T));
                    this->тестToolStripMenuItem1->Size = System::Drawing::Size(206, 22);
                    this->тестToolStripMenuItem1->Text = L"&Тест";
                    this->тестToolStripMenuItem1->Click += gcnew System::EventHandler(this,
&MyForm::тестToolStripMenuItem_Click);
                    //
                    // очиститиToolStripMenuItem1
                    //
                    this->очиститиToolStripMenuItem1->Image =
(cli::safe_cast<System::Drawing::Image^>(resources-
>GetObject(L"очиститиToolStripMenuItem1.Image")));
                    this->очиститиToolStripMenuItem1->Name = L"очиститиToolStripMenuItem1";
                    this->очиститиToolStripMenuItem1->ShortcutKeys =
static_cast<System::Windows::Forms::Keys>((System::Windows::Forms::Keys::Alt |
System::Windows::Forms::Keys::O));
                    this->очиститиToolStripMenuItem1->Size = System::Drawing::Size(206,
22);
                    this->очиститиToolStripMenuItem1->Text = L"&Очистити";
                    this->очиститиToolStripMenuItem1->Click += gcnew
System::EventHandler(this, &MyForm::очиститиToolStripMenuItem_Click);
                    //
                    // розрахуватиToolStripMenuItem1
                    //
                    this->розрахуватиToolStripMenuItem1->Image =
(cli::safe_cast<System::Drawing::Image^>(resources-
>GetObject(L"розрахуватиToolStripMenuItem1.Image")));
                    this->розрахуватиToolStripMenuItem1->Name =
L"розрахуватиToolStripMenuItem1";
                    this->розрахуватиToolStripMenuItem1->ShortcutKeys =
static_cast<System::Windows::Forms::Keys>(((System::Windows::Forms::Keys::Control |
System::Windows::Forms::Keys::Alt)
                            | System::Windows::Forms::Keys::C));
                    this->розрахуватиToolStripMenuItem1->Size = System::Drawing::Size(206,
22);
                    this->розрахуватиToolStripMenuItem1->Text = L"&Розрахувати";
                    this->розрахуватиToolStripMenuItem1->Click += gcnew
System::EventHandler(this, &MyForm::розрахуватиToolStripMenuItem_Click);
                    //
                    // toolStrip1
                    //
                    this->toolStrip1->Items->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^  >(7) {
                            this->toolStripButton_Тест,
                                    this->toolStripButton_Розрахунок, this-
>toolStripButton_Очищення, this->toolStripButton_Вихід, this->toolStripSeparator1, this-
>toolStripLabel_N,
                                    this->toolStripTextBox_N
                    });
                    this->toolStrip1->Location = System::Drawing::Point(0, 24);
                    this->toolStrip1->Name = L"toolStrip1";
                    this->toolStrip1->Size = System::Drawing::Size(491, 25);
                    this->toolStrip1->TabIndex = 2;
                    this->toolStrip1->Text = L"toolStrip1";
                    //
                    // toolStripButton_Тест
                    //
                    this->toolStripButton_Тест->Image =
(cli::safe_cast<System::Drawing::Image^>(resources-
>GetObject(L"toolStripButton_Тест.Image")));
```

```
                    this->toolStripButton_Тест->ImageTransparentColor =
System::Drawing::Color::Magenta;
                    this->toolStripButton_Тест->Name = L"toolStripButton_Тест";
                    this->toolStripButton_Тест->Size = System::Drawing::Size(51, 22);
                    this->toolStripButton_Тест->Text = L"Тест";
                    this->toolStripButton_Тест->Click += gcnew System::EventHandler(this,
&MyForm::тестToolStripMenuItem_Click);
                    //
                    // toolStripButton_Розрахунок
                    //
                    this->toolStripButton_Розрахунок->Image =
(cli::safe_cast<System::Drawing::Image^>(resources-
>GetObject(L"toolStripButton_Розрахунок.Image")));
                    this->toolStripButton_Розрахунок->ImageTransparentColor =
System::Drawing::Color::Magenta;
                    this->toolStripButton_Розрахунок->Name = L"toolStripButton_Розрахунок";
                    this->toolStripButton_Розрахунок->Size = System::Drawing::Size(94, 22);
                    this->toolStripButton_Розрахунок->Text = L"Розрахувати";
                    this->toolStripButton_Розрахунок->Click += gcnew
System::EventHandler(this, &MyForm::розрахуватиToolStripMenuItem_Click);
                    //
                    // toolStripButton_Очищення
                    //
                    this->toolStripButton_Очищення->Image =
(cli::safe_cast<System::Drawing::Image^>(resources-
>GetObject(L"toolStripButton_Очищення.Image")));
                    this->toolStripButton_Очищення->ImageTransparentColor =
System::Drawing::Color::Magenta;
                    this->toolStripButton_Очищення->Name = L"toolStripButton_Очищення";
                    this->toolStripButton_Очищення->Size = System::Drawing::Size(80, 22);
                    this->toolStripButton_Очищення->Text = L"Очистити";
                    this->toolStripButton_Очищення->Click += gcnew
System::EventHandler(this, &MyForm::очиститиToolStripMenuItem_Click);
                    //
                    // toolStripButton_Вихід
                    //
                    this->toolStripButton_Вихід->Image =
(cli::safe_cast<System::Drawing::Image^>(resources-
>GetObject(L"toolStripButton_Вихід.Image")));
                    this->toolStripButton_Вихід->ImageTransparentColor =
System::Drawing::Color::Magenta;
                    this->toolStripButton_Вихід->Name = L"toolStripButton_Вихід";
                    this->toolStripButton_Вихід->Size = System::Drawing::Size(55, 22);
                    this->toolStripButton_Вихід->Text = L"Вихід";
                    this->toolStripButton_Вихід->Click += gcnew System::EventHandler(this,
&MyForm::вихідToolStripMenuItem_Click);
                    //
                    // toolStripSeparator1
                    //
                    this->toolStripSeparator1->Name = L"toolStripSeparator1";
                    this->toolStripSeparator1->Size = System::Drawing::Size(6, 25);
                    //
                    // toolStripLabel_N
                    //
                    this->toolStripLabel_N->Name = L"toolStripLabel_N";
                    this->toolStripLabel_N->Size = System::Drawing::Size(10, 22);
                    this->toolStripLabel_N->Text = L" ";
                    //
                    // toolStripTextBox_N
                    //
                    this->toolStripTextBox_N->Name = L"toolStripTextBox_N";
```

```
this->toolStripTextBox_N->Size = System::Drawing::Size(100, 25);
//
// label_Kn
//
this->label_Kn->AutoSize = true;
this->label_Kn->Location = System::Drawing::Point(27, 60);
this->label_Kn->Name = L"label_Kn";
this->label_Kn->Size = System::Drawing::Size(77, 13);
this->label_Kn->TabIndex = 3;
this->label_Kn->Text = L"Коефіцієнт Kn";
//
// textBox_Kn
//
this->textBox_Kn->Location = System::Drawing::Point(30, 88);
this->textBox_Kn->Name = L"textBox_Kn";
this->textBox_Kn->Size = System::Drawing::Size(100, 20);
this->textBox_Kn->TabIndex = 4;
//
// textBox_n
//
this->textBox_n->Location = System::Drawing::Point(30, 146);
this->textBox_n->Name = L"textBox_n";
this->textBox_n->Size = System::Drawing::Size(100, 20);
this->textBox_n->TabIndex = 6;
//
// label_n
//
this->label_n->AutoSize = true;
this->label_n->Location = System::Drawing::Point(27, 118);
this->label_n->Name = L"label_n";
this->label_n->Size = System::Drawing::Size(133, 13);
this->label_n->TabIndex = 5;
this->label_n->Text = L"Швидкість мішалки, об/с";
//
// textBox_Rc
//
this->textBox_Rc->Location = System::Drawing::Point(221, 88);
this->textBox_Rc->Name = L"textBox_Rc";
this->textBox_Rc->Size = System::Drawing::Size(100, 20);
this->textBox_Rc->TabIndex = 8;
//
// label_Rc
//
this->label_Rc->AutoSize = true;
this->label_Rc->Location = System::Drawing::Point(218, 60);
this->label_Rc->Name = L"label_Rc";
this->label_Rc->Size = System::Drawing::Size(118, 13);
this->label_Rc->TabIndex = 7;
this->label_Rc->Text = L"Густина рідини, кг/м3";
//
// textBox_dm
//
this->textBox_dm->Location = System::Drawing::Point(221, 146);
this->textBox_dm->Name = L"textBox_dm";
this->textBox_dm->Size = System::Drawing::Size(100, 20);
this->textBox_dm->TabIndex = 10;
//
// label_dm
//
this->label_dm->AutoSize = true;
this->label_dm->Location = System::Drawing::Point(218, 118);
```

```
                    this->label_dm->Name = L"label_dm";
                    this->label_dm->Size = System::Drawing::Size(108, 13);
                    this->label_dm->TabIndex = 9;
                    this->label_dm->Text = L"Діаметр мішалки, м";
                    //
                    // MyForm
                    //
                    this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                    this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                    this->ClientSize = System::Drawing::Size(491, 178);
                    this->ContextMenuStrip = this->contextMenuStrip1;
                    this->Controls->Add(this->textBox_dm);
                    this->Controls->Add(this->label_dm);
                    this->Controls->Add(this->textBox_Rc);
                    this->Controls->Add(this->label_Rc);
                    this->Controls->Add(this->textBox_n);
                    this->Controls->Add(this->label_n);
                    this->Controls->Add(this->textBox_Kn);
                    this->Controls->Add(this->label_Kn);
                    this->Controls->Add(this->toolStrip1);
                    this->Controls->Add(this->menuStrip1);
                    this->MainMenuStrip = this->menuStrip1;
                    this->Name = L"MyForm";
                    this->StartPosition =
System::Windows::Forms::FormStartPosition::CenterScreen;
                    this->Text = L"Витрати потужності мішалки";
                    this->menuStrip1->ResumeLayout(false);
                    this->menuStrip1->PerformLayout();
                    this->contextMenuStrip1->ResumeLayout(false);
                    this->toolStrip1->ResumeLayout(false);
                    this->toolStrip1->PerformLayout();
                    this->ResumeLayout(false);
                    this->PerformLayout();

                }

#pragma endregion

private: System::Void тестToolStripMenuItem_Click(System::Object^  sender,
System::EventArgs^  e) {
       //Задавання значень змінних для тестового розрахунку
       textBox_Kn->Text = "0,32";
       textBox_n->Text = "3,96";
       textBox_Rc->Text = "1200";
       textBox_dm->Text = "2,8";
}
private: System::Void очиститиToolStripMenuItem_Click(System::Object^  sender,
System::EventArgs^  e) {
       //Очищення поля введення
       textBox_Kn->Text = "";
       textBox_n->Text = "";
       textBox_Rc->Text = "";
       textBox_dm->Text = "";
       toolStripLabel_N->Text = "";
       toolStripTextBox_N->Text = "";
}
private: System::Void розрахуватиToolStripMenuItem_Click(System::Object^  sender,
System::EventArgs^  e) {
       float Kn, n, Rc, dm, N;
       //Перетворює рядкове представлення числа в дійсне
       Kn = float::Parse(textBox_Kn->Text);
```

```
        n = float::Parse(textBox_n->Text);
        Rc = float::Parse(textBox_Rc->Text);
        dm = float::Parse(textBox_dm->Text);
        N = Kn*pow(n, 3)*Rc*pow(dm, 5);
        toolStripLabel_N->Text = "Витрати потужності на перемішування ";
        MessageBox::Show("Витрати потужності на перемішування = " + N.ToString("F"));
        toolStripTextBox_N->Text = N.ToString("F");
}
private: System::Void вихідToolStripMenuItem_Click(System::Object^  sender,
System::EventArgs^  e) {
        Close(); //Закрити програму
}
};
}
```

Завдання до комп'ютерного практикуму №3

Варіанти

1. Визначити втрату тиску P охолоджуючої води, яка протікає через свинцевий змійовик, який має 40 витків діаметром D = 0.5 м. Внутрішній діаметр свинцевої труби d = 0,012 м. Середня температура охолоджуючої води 20°C, а середня температура внутрішньої стінки труби 40 °C. Витрата охолоджуючої води G = 0,1 кг/с.

$$P = L \cdot \frac{l}{d} \cdot \frac{W^2}{2} \cdot p \cdot f \cdot f1,$$

де L = 0.0335 – коефіцієнт тертя;
l = 3.14*D*n – загальна довжина труби (n = 40), м;
$W = \frac{4 \cdot G}{3.14 \cdot p \cdot d^2}$ - швидкість руху рідини, м/с (з = 998 кг/м^3 – щільність води при 20°C);
f, f1 – поправкові множини для неізотермічного потоку рідини у непрямих трубах (f = 0.85; f1 = 1+ 3,54d/D).

2. Знайти коефіцієнт тепловіддачі L при конденсації пару на поверхні труби довжиною H-2м, яка утворює з горизонтальною поверхнею кут b= $\frac{П}{3}$. Температура стінки $t_{ст} = 210\ ^0C$, температура пари $t \quad = 220\ ^0C$.

$$L_b = L\sqrt{\sin b},$$

де L – коефіцієнт тепловіддачі для вертикальної труби, який розраховується за формулою:

$$L = 3 \cdot 10^{-3} \sqrt{\frac{H(t - t_{ст}) \cdot l^3 \cdot p^2 \cdot g}{r \cdot m^3}}$$

тут l = 0,652 Вт/(м*К) – теплопровідність конденсату;
p = 846 кг/м^3 - щільність конденсату;
g = 9,81 м/с^2 - прискорення вільного падіння;
r = 1,88*10^6 Дж/кг – теплота пароутворення;
m = 1,29*10^{-4} н*с/м^2 - динамічна в'язкість конденсату.

3. Розрахувати константу швидкості реакції між йодистим метилом і диметил-р-толундіном у розчині нітробензолу.

$$\frac{\frac{x + x_p}{a}}{a \cdot t \cdot \left[1 - \left(\frac{x_p}{a}\right)^2\right]} \cdot \ln \left| \frac{\frac{x}{a}\left(1 - \frac{x + x_p}{a^2}\right) - 1}{a^2 \cdot \left(\frac{x_p}{a} - \frac{x}{a}\right)} \right|$$

де a = 0,06 – початкова концентрація;
x = 0,01246 - рівноважна концентрація;
x_p = 0,00675 – поточна концентрація;
$t = 12.$

4. Розрахувати в'язкість діоксину сірки при 300 ^0C і атмосферному тиску, використавши формулу:

$$\mu = 6{,}3 \cdot 10^{-4} \cdot \frac{м^{1/2} \cdot P_{кр}^{2/3}}{T_{кр}^{1/6}} \cdot \frac{T_{прив}^{3/2}}{T_{прив} \cdot 0{,}8}$$

де м = 64 – молекулярна вага SO_2;
$T_{кр}$ = 430 ^0C - критична температура;
$P_{кр}$ = 77,7 атм – критичний тиск;
$T_{прив}$ = (300 + 273)/430 – приведена температура.
Надрукувати значення μ та $\sqrt{\mu}$.

5. Рідина витікає із прямої посудини 1 до посудини 2 через круглий отвір *d* = 2 см = 0,02 м. Різниця рівнів води в початковий момент *H* = 2м. Площина поперечних перерізів посудини S_1 = 8м^2, S_2 = 6м^2. Знайти час, необхідний для того, щоб різниця рівнів зменшилася до 1м, для цього використати формулу:

$$t = \frac{8 \cdot S_1 \cdot S_2}{c \cdot 3.14 \cdot d^2 (S_1 + S_2) \cdot \sqrt{2 \cdot g}} \left(\sqrt{H_0} - \sqrt{H}\right)$$

де *c* = 0,62 – коефіцієнт витрат; *g* = 9.81
надрукувати значення t, e^c, t^2, $sin^2 d$.

6. Розрахувати коефіцієнт ефективності, використовуючи формулу:

$$\Phi_s = \frac{r_s}{3} \sqrt{\frac{W_p \cdot p_k}{n \cdot D_0 \cdot (C_s - C_0)}}$$

де $r_s = 0{,}317$;
W_p = 4,85*10^{-4};
p_k = 1;
$n = 0{,}93$;
$D_0 = 0{,}08$;
$C_s = 1{,}6 \cdot 10^{-4}$;
$C_0 = 0$.
Надрукувати значення Φ_s, e^n, $sin_s r$.

7. Визначити константу швидкості реакції окислення оксиду азоту, використовуючи формулу:

$$K = \frac{1}{t} \cdot \frac{1}{(b-a)^2} \cdot \left[\frac{(b-a) \cdot x}{(a-x) \cdot a} + \ln \frac{(a-x) \cdot b}{(b-x) \cdot a}\right]$$

де b=7,06;
a=5,4;
t=11;
x=2,25.

8. Знайти діаметр корпусу випарювального апарату з природною циркуляцією розчину, використовуючи формулу:

$$D = \sqrt{\frac{1{,}27 \cdot n \cdot t^2 \cdot \sin L}{f} + (d \cdot 2 \cdot t)^2}$$

де n=1210 – кількість трубок;
t=0,05м;
d=0,4м;
f=0,85м;

L=1,047рад.

9. Знайти коефіцієнт дифузії молекул ізобутану через шар зернин оксиду алюмінію, використовуючи формулу:

$$D = \frac{-R \cdot T \cdot r \cdot g_a}{0{,}023 \cdot P \cdot (x_2 - x_1)}$$

де R = 82,298 – універсальна газова стала;
T = 0,31;
r = 0,5;
g_a = 0,08*10^{-6};
P = 5;
x_2 = 0;
x_1 = 1.
Надрукувати значення D, $\sqrt{D}$, e^T.

10. Визначити коефіцієнт дифузії двооксиду сірки у повітрі $t = 20\ ^0C$ та P=1атм. Використати формулу:

$$D = 4{,}3 \cdot 10^{-3} \cdot \frac{T^{3/2}}{P \cdot (V_A^{1/3} - V_B^{1/3})^2} \cdot \sqrt{\frac{1}{M_A} + \frac{1}{M_B}}$$

де $V_A - V_{SO_2}$ = 44,8 $см^3/моль$;
$V_B - V_{ВОЗ}$ = 29,9 $см^3/моль$;
T = 273 +t;
M_A = 64;
M_B = 28,95.

11. Визначити теплопровідність рідкого гептану при температурі $t = 20\ ^0C$, використовуючи формулу:

$$L = A \cdot C_M \cdot y_{відн} \cdot \sqrt[3]{\frac{y_{отн}}{M}}$$

де A = 1,52;
C_M = 51,9 + 0,142 * (t + 30), кал/(моль * град) – мольна теплоємність;
$y_{відн}$ = 0,675 – відносна питома вага;
M = 100;
C = $C_M/100$ - питома теплоємність.

12. Знайти тривалість адсорбції суміші парів етилового спирту та діетилового ефіру шаром активованого вугілля висотою H = 1м, використовуючи формулу:

$$t = \frac{a_0}{w \cdot C_0} \cdot \left[H - \frac{w}{b} \cdot (\ln\frac{C_0}{C} - 1)\right]$$

де a_0 = 100кг/$м^3$ - кількість адсорбції речовини, зрівноваженої з концентрацією потоку;
$w = 12м/хв$ - швидкість парогазового потоку;
C_0 = 0,072кг/$м^3$ – початкова концентрація;
C = 0,001кг/$м^3$ – середня концентрація на виході;
b = 60 * 7,4 $хв^{-1}$ - коефіцієнт масо передачі.

13. Розрахувати в'язкість аміаку в інтервалі температур від 12 до 18°С через 1°С за формулою Сатерленда

$$N = N_0 \frac{273{,}15 + C}{T \cdot C} \cdot (T/273{,}15)^n$$

При умові, що множник n визначається так:

$$n = A - B \cdot (t - 273{,}15) \cdot D \cdot 10^{-7} \cdot (t - 273{,}15)^2$$

де *A* = 1,06; *B* = 1,04; *D* = 0; *C* = 503; *N* = 91,6 * 10^{-7} Па*с.

14. Розрахувати ізобарну теплоємність CO_2 в інтервалі температур від 30 до 150°С через 10°С за формулою:

$$C_p = E + F \cdot (T/100) + G \cdot (T/100) + H \cdot (T/100) + N \cdot (T/100)$$

де *E* = 0,81513; *F* = 9,8454· 10^{-2}; *G* = -9,4747· 10^{-3}; *H* = 36,006· 10^{-5}; *N* = -0,0567.

15. Використавши формулу завдання 2 при *E* = 2,0183; *F* = 158,072 * 10^{-2}; *G* = 273,61 * 10^{-3}; *H* = -9380,9 * 10^{-5}; *N* = -1,9986, розрахувати ізобарну теплоємність аміаку в інтервалі температур від 80 до 150°С через 10°С.

15. За умови завдання 2 розрахувати ізохорну теплоємність CO_2. Формула зв'язку має вигляд: $C_v = C_p - R/M$ (*C* – молекулярна маса газу; *R* – універсальна газова стала).

17. За умови завдання 2 розрахувати ізохорну теплоємність аміаку. Формула зв'язку має вигляд: $C_v = C_p - R/M$ (C – молекулярна маса газу; R – універсальна газова стала).

18. Розрахувати в'язкість води в інтервалі температур від 16 до 27°С через 1°С за формулою:

$$N_o = A(B + t)^n,$$

де *A* = 0,59849, *B* = 43, 252, *n* = -1,5423.

18. Розрахувати теплопровідність аміаку в інтервалі температур від 16 до 27°С через 1°С за формулою:

$$L = A \frac{(T_K + c)}{(T + C)} \cdot \frac{T^{\frac{3}{2}}}{T_K^{\frac{3}{2}} M^{\frac{5}{6}}},$$

де A = 4,3643 * 10^{-3}; C = 503; T_K = 239,73 К; *M* – молекулярна маса газу.

20. Розрахувати густину повітря в інтервалі температур від 45 до 54°С за формулою:

$$R = \frac{1{,}293 \cdot P}{(1 + 0{,}00367 \cdot t) \cdot 760}, \text{кг/м}^3$$

де *P* – тиск, який дорівнює 105,2 кПа.

21. Розрахувати швидкість осадження кулеподібної частинки радіусом r ($r = 0{,}2 \cdot 10^{-3}$ м, $r = 0{,}3 \cdot 10^{-3}$ м … $r = 0{,}7 \cdot 10^{-3}$ м) у газі. Використати формулу:

$$W_{oc} = \frac{d^2 \cdot R \cdot g}{18 \cdot M_c},$$

де g = 9,81 м^2/с, R = 1684 кг/м^3; M_c = 5,2 * 10^{-4} Н * с/м^2.

22. Розрахувати час процесу фільтрування суспензії об'ємом V ($V = 50 \cdot 10^{-6}$, $100 \cdot 10^{-6}$, $150 \cdot 10^{-6}$, $200 \cdot 10^{-6}$, $250 \cdot 10^{-6}$ м^3) за формулою:

$$t = \frac{M \cdot r_0 \cdot x_0}{2p} \cdot \frac{V^2}{S^2} + \frac{M \cdot R}{P} \cdot \frac{V}{S},$$

де $M = 5{,}17 \cdot 10^{-9}$ Н * с/м^3; $r = 4{,}15 * 10^{12}$ м$^{-2}$; $R = 2{,}3 * 10^{10}$ м$^{-1}$; $x_0 = 0{,}178$ м^3/м^3; $P = 0{,}6 \cdot 10^5$ Н/м^2; $S = 1$ м.

23. Розрахувати витрату потужності на перемішування для пропелерної мішалки діаметром d = [2,6; 2,8; 3,0; 3,2 м]. Використати формулу

$$N = K_N \cdot n^3 \cdot R_C \cdot d_m^5,$$

де K_N = 0,32; т = 3,96 об/с; R_C = 1200 кг/м^3.

24. Розрахувати коефіцієнт теплопровідності L нітробензолу в інтервалі температур від 40 до 110°С через 10°С. Використати формулу

$$L_t = L_{30} \cdot [1 - E \cdot (t - 30)],$$

де $L_{30} = \mathrm{A}_1 \cdot \mathrm{c} \cdot R \cdot \sqrt{\frac{R}{M}}$, $\mathrm{A}_1 = 4{,}22 \cdot 10^{-2}$; с=1,38· 10^3 Дж/(кг*град); R = 1200 кг/м^3; M = 123; E =1· 10^{-3} град$^{-1}$.

25. Розрахувати значення густини та в'язкості сірчаної кислоти в інтервалі температур від 10 до 80°С через 10°С. Використати формулу

$$R = 1894{,}8 - 0{,}909 \cdot t; \qquad M = 1{,}406 \cdot 10^{-3} + 5{,}0087 \cdot 10^{-1}/t$$

26. Розрахувати опір шару осадка при фільтруванні суспензії з діаметром частинок $d_{\text{ср}}$ = (0,1; 0,15; 0,20; 0,25) * 10^{-3} м за формулою:

$$r_{0_{11}} = A \cdot d_{cp}^B,$$

де A = 9,8373*10; B = -0,3224.

27. Розрахувати тиск пару хлористого метилену P (та $\ln P$) в інтервалі температур 20…40°С через 5°С, якщо

$$\ln P = -\frac{A}{T} + B \cdot \ln T + C,$$

де A = 1995,6; B = -3,4426; $T = t + 273$; C = 8,321; 8,621; 8,921.

28. Розрахувати швидкість R хімічної реакції A - > B в діапазоні температур t = 300…400К з кроком 20 К, якщо

$$R = \mathrm{K}_0 \cdot \exp\left(-\frac{E}{R} \cdot T\right) \cdot C_0,$$

де K_0 = 2,05 * с$^{-1}$; R = 1,98725 кал/моль*град; E = 20100 кал/моль; C_0 = (2; 4; 6) моль/м^3.

29. Розрахувати, який об'єм займе 10 г азоту при тиску *P* = 1,5; 3,0; 4,5 атм і зміні температури від 200 до 400 К з кроком 10 К.
(Використати формулу 1; *R* = 0,082057 л*атм/моль*град)

30. Розрахувати значення константи швидкості хімічної реакції

$$K = A \cdot \exp\left(\frac{-5{,}29 \cdot 10^4}{R \cdot T}\right)$$

В інтервалі температур 600 … 800 через 50°С, якщо А = $7{,}5 \cdot 10^{-5}$; $8{,}0 \cdot 10^{-5}$; $8{,}5 \cdot 10^{-5}$.

Контрольні питання

1) Як додати об’єкт головного меню у форму? Які елементи може містити головне меню? В чому їх призначення?
2) Як додати об’єкт панелі інструментів у форму? Які елементи може містити панель інструментів? В чому їх призначення?
3) Перерахуйте основні властивості пунктів меню та їх призначення.
4) Перерахуйте основні властивості елементів панелі інструментів та їх призначення.
5) Елементами якого класу є пункти меню? Як додати обробник подій до пункту меню?
6) Чи може форма містити більше одного об’єкту головного меню? Якщо так, то для чого це потрібно?
7) Як створити контекстне меню? Як організувати появу різних контекстних меню для різних візуальних об’єктів форми?

Комп'ютерний практикум №4

Табличне введення даних

Мета: вивчити прийоми роботи з таблицями з використанням об'єкту DataGridView. Засвоїти методи створення, збереження та відкриття таблиць у форматі файлу XML. Ознайомитись з роботою об'єкту DataSet у проектах Windows Forms у Visual C++. Навчитися створювати об'єкти DataTable для збереження даних у табличному вигляді. Ознайомитись з прийомами побудови графіків з використанням об'єкту Chart.

Завдання: створити новий пустий проект CLR, додати в нього форму Windows Forms. Додати необхідні об'єкти з Панелі елементів для створення головного та контекстного меню, а також панелі інструментів. Додати до форми об'єкт DataGridView для відображення таблиці результатів у формі. Додати об'єкт Chart для створення графіків на основі табличних даних. Створити об'єкти потрібних класів для реалізації поставленої задачі згідно отриманого варіанту завдання. Передбачити можливість збереження таблиці даних у обраний користувачем файл формату XML. Надати користувачу можливість відкривати набір даних з вже існуючого файлу формату XML. Організувати побудову графіків на основі стовпців створеної таблиці з використанням об'єкту Chart.

Загальні вимоги.

1) Створити проект Windows Forms.
2) Додати в нього необхідні елементи керування та програмний код для реалізації поставленої задачі згідно отриманого варіанту завдання
3) Програмний код має бути чітко структурований.
4) Імена об'єктів мають нести сенсові навантаження.
5) Програмний код має супроводжуватись коментарями в тексті програми.

Вимоги до виконання.

1) Створити новий пустий проект CLR в Visual C++ з використанням створеного шаблону проектів Windows Forms з ім'ям виду Прізвище_КПР4.
2) Додати у форму об'єкт головного меню класу MenuStrip.
3) Додати у форму об'єкт контекстного меню класу ContextMenuStrip.
4) Додати у форму об'єкт панелі інструментів класу ToolStrip.
5) Додати об'єкт DataGridView для відображення таблиці даних.
6) Додати інші необхідні об'єкти для створення програмного інтерфейсу та запрограмувати їх роботу.

7) Передбачити збереження табиці даних у зовнішньому файлі.
8) Передбачити можливість відкриття збереженого файлу таблиці в об'єкті DataGridView вікна додатку.
9) Додати у форму об'єкт Chart та створити за його допомогою графіки у вікні додатка на основі табличних даних.

Теоретичні відомості

4.1 Введення табличних даних

Існує безліч завдань, які передбачають введення даних у вигляді таблиць. Звичайно, можна цю таблицю програмувати як сукупність текстових полів TextBox, але в більшості випадків заздалегідь невідомо, скільки рядів даних буде вводити користувач, необхідно передбачити скролінг цієї таблиці і т. д. Тобто проблем в організації введення табличних даних досить багато.

Ми пропонуємо для введення табличних даних використовувати елемент управління DataGridView (Сітка даних). Перш за все, цей елемент управління призначений для відображення даних, які зручно представити у вигляді таблиці, найчастіше джерелом цих даних є база даних. Однак, крім відображення, елемент управління DataGridView дозволяє також редагувати табличні дані. DataGridView підтримує виділення, зміну, видалення, розбиття на сторінки і сортування.

Приклад програмної реалізації

Приклад створення проекту Windows Forms для відображення даних у табличному вигляді та збереженя даних у файлі XML.

Програма, що розглядається в даному прикладі, пропонує вам заповнити таблицю телефонів знайомих, співробітників, родичів, коханих і т. д. Якщо клацнути на кнопці Запис дана таблиця записується на диск у файл у форматі XML. Для спрощення тексту програми передбачений запис в один і той же файл D:\tabl.xml. При наступних запусках даної програми таблиця буде зчитуватися з файлу, і ви зможете продовжити редагування таблиці. Тому цю програму можна голосно назвати «табличним редактором». Клацаючи на заголовках колонок, можна розташувати записи в колонках в алфавітному порядку для зручного пошуку необхідного телефону.

Для написання програми запустимо Visual Studio і в вікні New Project виберемо в середовищі CLR вузла Visual C++ додаток шаблону Windows Forms середовища Visual C++. Далі потрібно з панелі управління Toolbox перенести мишею наступні елементи управління: сітку даних DataGridView і кнопку Button. Текст програми приведений в наступному лістингу.

```
#pragma endregion
	// Програма пропонує користувачеві заповнити таблицю телефонів його
	// Знайомих, співробітників, родичів, коханих і т. д. Якщо клацнути
	// На кнопці Запис дана таблиця записується на диск у файл у форматі
	// XML. Для спрощення тексту програми передбачений запис в один і той
	// Же файл C: \ tabl.xml. При наступних запусках даної програми таблиця
	// Буде зчитуватися з цього файлу, і користувач може продовжувати
	// Редагування таблиці
	// ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
	DataTable^ Таблиця; // Об'явлення объекта "Таблиця"
	DataSet^ НабірДаних; // Об'явлення объекта "НабірДаних"
	private: System::Void Form1_Load(System::Object^ sender, System::EventArgs^ e)
			{
				this->Text = "Простий табличний редактор";
				button1->Text = "Запис";
				Таблиця = gcnew DataTable();
				НабірДаних = gcnew DataSet();
				if (IO::File::Exists("D:\\tabl.xml") == false)
				{
					// Якщо XML-файлу НЕМАЄ:
					dataGridView1->DataSource = Таблиця;
					// Заповнення "шапки" таблиці
					Таблиця->Columns->Add("Імена");
					Таблиця->Columns->Add("Номери телефонів");
					// Додати об'єкт Таблиця в DataSet
					НабірДаних->Tables->Add(Таблиця);
				}
				else // Якщо XML-файл Є:
				{
					НабірДаних->ReadXml("D:\\tabl.xml");
					// Вміст DataSet у вигляді рядка XML для відладки:
					String ^ РядокXML = НабірДаних->GetXml();
					dataGridView1->DataMember = "Назва таблиці";
					dataGridView1->DataSource = НабірДаних;
				}
			}
	private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)
			{
				// Зберегти файл tabl.xml:
				Таблиця->TableName = "Назва таблиці";
				НабірДаних->WriteXml("D:\\tabl.xml");
			}
```

Як видно з тексту програми, треба було всього лише кілька рядків програмного коду для створення такої багатофункціональної програми. Це стало можливим завдяки використанню потужної сучасної технології ADO.NET. На початку класу оголошені два об'єкти цієї технології: набір даних DataSet і таблиця даних DataTable. Об'єкт класу DataSet є основним компонентом архітектури ADO.NET. DataSet представляє кеш даних, розташований в оперативній пам'яті. DataSet складається з колекції об'єктів класу DataTable. Тобто в один об'єкт класу DataSet може входити кілька таблиць, а інформацію про них ми можемо записувати в файл на диск одним оператором WriteXml, відповідно читати - ReadXML. Таким чином, в цій програмі ми маємо справу переважно з трьома об'єктами: DataSet - кеш даних, DataTable - представляє

одну таблицю з даними та DataGridView - елемент управління для відображення даних.

Відразу після ініціалізації компонентів форми ми обробили дві ситуації. Якщо файлу, в який ми зберігаємо інформацію про таблиці, не існує Exists («D: \\ tabl.xml») == false, то призначаємо в якості джерела даних DataSource для DataGrid об'єкт класу DataTable і заповнюємо «шапку» таблиці, тобто вказуємо назви колонок: «Імена» і «Номери телефонів», а потім додаємо об'єкт DataTable в об'єкт DataSet. Тепер користувач бачить порожню таблицю з двома колонками і може почати її заповнення. Якщо файл існує (гілка else), то дані в об'єкт DataSet відправляємо з XML-файла (ReadXML). Тут уже як джерело даних для сітки даних DataGrid вказуємо об'єкт DataSet.

При натисканні мишею на кнопці Запис (рис. 4.1) - подія button1.Click - відбувається запис XML-файла на диск (WriteXml).

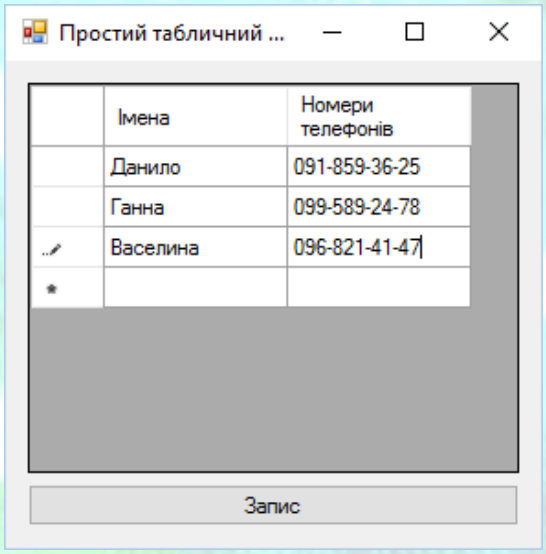

Рисунок 4.1 – Вікно програми з введеними даними

Тут використовуються, так звані, XML-файли. Формат цих файлів дозволяє легко і надійно передавати дані за допомогою Інтернету навіть на комп'ютери іншої платформи (наприклад, Macintosh). Ми не ставили перед собою мету працювати в Інтернеті, а всього лише скористалися цією технологією. Файл формату XML можна переглянути Блокнотом або за допомогою MS Word, оскільки це текстовий файл. Однак слід врахувати, що цей файл записаний в кодуванні UTF-8, тому іншими текстовими редакторами, наприклад edit.com або Rpad32.exe (російський Блокнот), його прочитати важко. XML-документ відкривається веб-браузером, XML-editor (входить до складу Visual Studio), MS Office SharePoint Designer, MS Front Page і іншими програмами. При цьому застосування відступів і різних колірних рішень дозволяє більш наочно продемонструвати структуру даного файлу. XML-файл можна відкрити

табличним редактором MS Excel, і при цьому він може відобразитися у вигляді таблиці. На рис. 4.2 наведено зразок подання XML-файла в браузері.

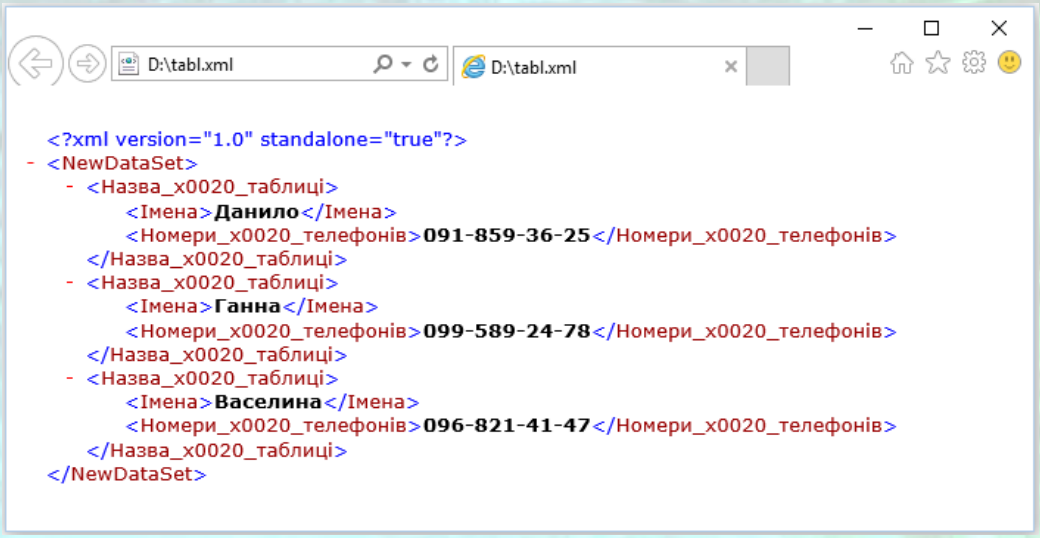

Рисунок 4.2 – Файл XML у вікні браузера

При використанні нами XML-файлів для програмування найпростішого табличного редактора зовсім не обов'язково вникати в його структуру, тим більше при виборі імені файлу для збереження зовсім не обов'язково встановлювати розширення файлу xml, файл з будь-яким розширенням буде читатися методом ReadXml як XML-файл.

Програміст може отримати доступ до полів таблиці. Наприклад, доступ до лівої верхньої комірки (поля) таблиці можна отримати, використовуючи властивість об'єкта класу DataTable: Таблiця.Rows.Item(0).ltem(0). Однак запис цього поля, наприклад, в послідовний файл буде некоректним навіть при використанні додаткової змінної через те, що технологія ADO.NET передбачає кешування даних. Таким чином, читання і запис даних для подібних таблиць слід організовувати тільки через методи об'єкта DataSet.

Відзначимо, що дана програма може також бути інструментом для створення XML-файлів. Текст програмного модуля наведений в лістингу 4.1.

4.2 Побудова графіку з використанням елементу Chart

На Панелі елементів (Toolbox) знайдемо елемент Chart. Він призначений для виведення в екранну форму графіка (діаграми). Дуже зручно будувати цей графік з табличних даних, представлених у вигляді об'єкта класу DataTable.

У прикладі наведеному нижче вирішимо наступну задачу. Дано відомості про обсяги продажів за п'ять місяців. Потрібно наочно візуалізувати ці дані на графіку у вигляді гістограми.

В межах програми також передбачимо можливість зміни типу діаграми. Для цього застосуємо елемент ComboBox з якого користувач зможе обирати потрібний тип діаграми.

Приклад програмної реалізації

Приклад використання елементів керування Chart і DataGridView, виведення графіку (діаграми) залежнсті обсягів продажів від часу за місяцями.

Для вирішення цього завдання запустимо Visual Studio і в вікні Створення проекту (New Project) виберемо в середовищі CLR вузла Visual C++ додаток шаблону Windows Forms. З Панелі елементів (ToolBox) перенесемо в проектовану екранну форму наступні елементи: діаграму Chart, сітку даних DataGridView та спадаючий перелік рядків ComboBox. У лістингу приведений програмний код вирішення задачі.

```
#pragma endregion
            // Програма використовує елементи керування Chart і DataGridView, виводить
            // графік (діаграму) залежнсті обсягів продажів від часу за місяцями.
            // При цьому в якості джерела даних вказуємо об'єкт класу DataTable
            // Стиль діаграми можна міняти за допомогою переліку з comboBox1
private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
            this->Text = "Побудова діаграми";
            DataTable ^ Таблиця = gcnew DataTable();
            // У цій таблиці замовляємо дві колонки "Місяць" і "Обсяг продажів":
            Таблиця->Columns->Add("Місяць", String::typeid);
            // У C #: Табліца.Columns.Add("Місяць", typeof(String));
            // Значення в другій колонці призначаємо типу long:
            Таблиця->Columns->Add("Обсяг продажів", long::typeid);
            // У C #: Табліца.Columns.Add("Обсяг продажів", typeof(long));
            // Заповнити перший рядок (ряд) в таблиці:
            DataRow ^ Ряд = Таблиця->NewRow();
            Ряд["Місяць"] = "Травень"; Ряд["Обсяг продажів"] = 15;
            Таблиця->Rows->Add(Ряд);
            // Додаємо другий рядок в таблиці:
            Ряд = Таблиця->NewRow();
            Ряд["Місяць"] = "Червень"; Ряд["Обсяг продажів"] = 35;
            Таблиця->Rows->Add(Ряд);
            // Додаємо третій рядок:
            Ряд = Таблиця->NewRow();
            Ряд["Місяць"] = "Липень"; Ряд["Обсяг продажів"] = 65;
            Таблиця->Rows->Add(Ряд);
            // Додаємо четвертий рядок:
            Ряд = Таблиця->NewRow();
            Ряд["Місяць"] = "Серпень"; Ряд["Обсяг продажів"] = 85;
            Таблиця->Rows->Add(Ряд);
            // Додаємо п'ятий рядок:
            Ряд = Таблиця->NewRow();
            Ряд["Місяць"] = "Вересень"; Ряд["Обсяг продажів"] = 71;
            Таблиця->Rows->Add(Ряд);

            // Наведені нижче властивості можна задати у вікні Властивості
            // для об'єкту chart1
```

```
            // Складену таблицю вказуємо як джерело даних:
            chart1->DataSource = Таблиця;
            // На одному графіку можна зобразити кілька залежностей.
            // Наприклад, перша залежність - обсяги продажів за вказаними
            // Місяців в 2014 році, і друга залежність - продажу по
            // тим же місяцям в 2015 році.
            // В даному графіку ми покажемо тільки одну залежність, дані
            // для відображення цієї залежності назвемо "Series1"
            // На горизонтальній осі відкладаємо назви місяців:
            chart1->Series["Series1"]->XValueMember = "Місяць";
            // А по вертикальній осі відкладаємо обсяги продажів:
            chart1->Series["Series1"]->YValueMembers = "Обсяг продажів";
            // Назва графіка (діаграми):
            chart1->Titles->Add("Обсяг продажів за місяцями");
            // Задаємо тип діаграми - стовпчикова гістограма:
            chart1->Series["Series1"]->ChartType = System::Windows::Forms::
                  DataVisualization::Charting::SeriesChartType::Column;
            // Тип діаграми може бути іншим, наприклад: Pie, Line і ін.
            chart1->Series["Series1"]->Color = Color::Aqua;
            // Легенду на графіку не відображаємо:
            chart1->Series["Series1"]->IsVisibleInLegend = false;
            // Прив'язка графіка до джерела даних:
            chart1->DataBind();
            // Для сітки даних вказати джерело даних
            dataGridView1->DataSource = Таблиця;

            // Задаємо перелік типів діаграм у об'єкті comboBox1
            // Це можна задати у вікні властивостей comboBox1 в полі Items
            comboBox1->Items->Add("Точкова");              //Point
            comboBox1->Items->Add("Графік");               //Line
            comboBox1->Items->Add("Згладжений графік"); //Spline
            comboBox1->Items->Add("Секторна");                    //Pie
            comboBox1->Items->Add("Пелюсткова");           //Radar
            comboBox1->Items->Add("Гістограма");           //Bar
            comboBox1->Items->Add("Стовпчаста");           //Column
            comboBox1->Text = "Оберіть тип діаграми";

            // Встановлюємо вирівнювання об'єктів в межах форми
            chart1->Dock = System::Windows::Forms::DockStyle::Top;
            dataGridView1->Dock = System::Windows::Forms::DockStyle::Bottom;
            chart1->Height = this->Height - dataGridView1->Height - 40;
            comboBox1->Left = 0;
            comboBox1->Top = 0;
      }

private: System::Void comboBox1_SelectedIndexChanged(System::Object^  sender,
System::EventArgs^  e) {
      // Змінюємо тип діаграми в залежності від обраного рядка в об'єкті  comboBox1
      switch (comboBox1->SelectedIndex)
            {
            case 0:
                  chart1->Series["Series1"]->ChartType =
System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Point;
                  break;
            case 1:
                  chart1->Series["Series1"]->ChartType =
System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Line;
                  break;
            case 2:
                  chart1->Series["Series1"]->ChartType =
System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Spline;
```

```
                        break;
                case 3:
                        chart1->Series["Series1"]->ChartType =
System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Pie;
                        break;
                case 4:
                        chart1->Series["Series1"]->ChartType =
System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Radar;
                        break;
                case 5:
                        chart1->Series["Series1"]->ChartType =
System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Bar;
                        break;
                default:
                        chart1->Series["Series1"]->ChartType =
System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Column;
                        break;
                }
        }

private: System::Void MyForm_Resize(System::Object^  sender, System::EventArgs^  e) {
        // Підбір висоти об'єкту chart1 при зміні розміру вікна
        chart1->Height = this->Height - dataGridView1->Height - 40;
}
};
```

Значну частину з наведеного коду можна скоротити шляхом задавання значень властивостей у вікні Властивості (Propeties) під час проектування інтерфейсу. Це спрощує процес створення інтерфейсу і не вимагає ручного написання коду. Під час розробки інтерфейсу проекту потрібно максимально використовувати можливості вікна Властивості (Propeties). В наведеному коді всі властивості задаються та змінюються програмно для можливості швидкого відтворення прикладу шляхом простого вставлення вище наведеного коду у новий проект.

При створенні даної програми використано три події, реакцією на які є наведений програмний код: події Load та Resize форми і подія SelectedIndexChanged об’єкту ComboBox.

У програмному коді при обробці події завантаження форми оголошуємо об'єкт Таблиця класу DataTable. Цей об'єкт представляє одну таблицю даних в оперативній пам'яті. Щоб візуалізувати цю таблицю на екрані, використовують елемент - сітка даних DataGridView. Об'єкт класу DataTable використовують в якості вихідних даних і для сітки даних DataGridView, і для діаграми Chart.

У таблиці DataTable визначаємо її схему, замовляючи дві колонки «Місяць» та «Обсяг продажів». А далі заповнюємо таблицю по її рядках, використовуючи метод Add. Заповнену п'ятьма рядками таблицю вказуємо як джерело даних для елементів Chart і DataGridView. Далі оформляємо зовнішній вигляд діаграми, що детально представлено в коментарях до програмного коду.

Фрагмент роботи програми показаний на рис. 4.3.

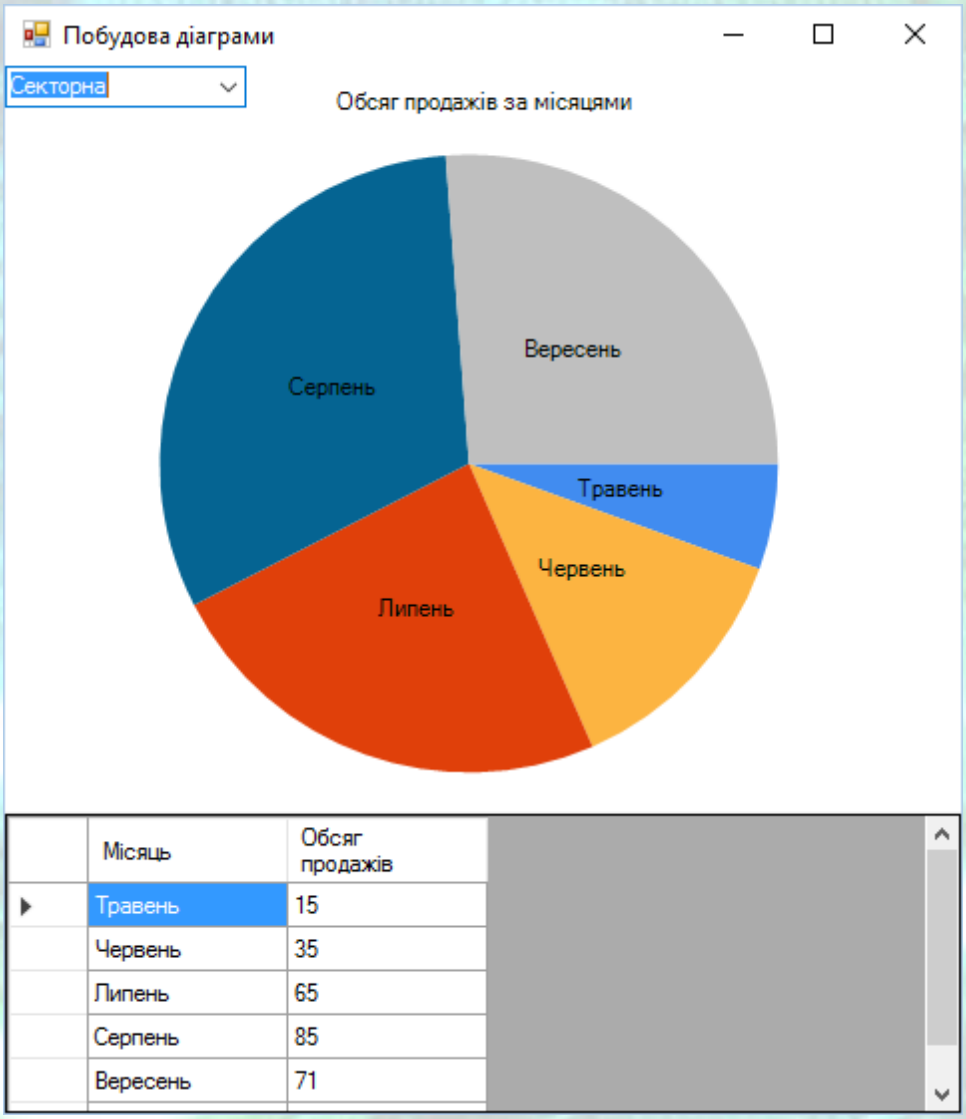

Рисунок 4.3 – Побудова діаграм

В об’єкті ComboBox у вікні запущеного доданку можна зі спадаючого списку обрати один з семи передбачених в програмі типів діаграм. Після обрання типу діаграми виникає подія `SelectedIndexChanged`, в якій здійснюється зміна типу діаграми.

Повний текст програмного модуля наведений в лістингу 4.2.

Програмний код

Лістинг 4.1

```
//Програма ілюстрації роботи з табличними даними

#pragma once

namespace ТаблВвод {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Сводка для Form1
	/// </summary>
	public ref class Form1 : public System::Windows::Forms::Form
	{
	public:
		Form1(void)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
		}

	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~Form1()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::DataGridView^  dataGridView1;
	protected:
	private: System::Windows::Forms::Button^  button1;

	private:
		/// <summary>
		/// Требуется переменная конструктора.
		/// </summary>
		System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
		/// <summary>
		/// Обязательный метод для поддержки конструктора - не изменяйте
		/// содержимое данного метода при помощи редактора кода.
		/// </summary>
		void InitializeComponent(void)
		{
			this->dataGridView1 = (gcnew System::Windows::Forms::DataGridView());
```

```
                    this->button1 = (gcnew System::Windows::Forms::Button());
                    (cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(this-
>dataGridView1))->BeginInit();
                    this->SuspendLayout();
                    //
                    // dataGridView1
                    //
                    this->dataGridView1->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;
                    this->dataGridView1->Location = System::Drawing::Point(12, 12);
                    this->dataGridView1->Name = L"dataGridView1";
                    this->dataGridView1->Size = System::Drawing::Size(268, 213);
                    this->dataGridView1->TabIndex = 0;
                    //
                    // button1
                    //
                    this->button1->Location = System::Drawing::Point(12, 231);
                    this->button1->Name = L"button1";
                    this->button1->Size = System::Drawing::Size(268, 23);
                    this->button1->TabIndex = 1;
                    this->button1->Text = L"button1";
                    this->button1->UseVisualStyleBackColor = true;
                    this->button1->Click += gcnew System::EventHandler(this,
&Form1::button1_Click);
                    //
                    // Form1
                    //
                    this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                    this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                    this->ClientSize = System::Drawing::Size(292, 266);
                    this->Controls->Add(this->button1);
                    this->Controls->Add(this->dataGridView1);
                    this->Name = L"Form1";
                    this->Text = L"Form1";
                    this->Load += gcnew System::EventHandler(this, &Form1::Form1_Load);
                    (cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(this-
>dataGridView1))->EndInit();
                    // ..............
                    // Программный код, расположенный выше, создан средой Visual Studio
                    // автоматически, поэтому автором не приводится
                    this->ResumeLayout(false);
        }
#pragma endregion
            // Програма пропонує користувачеві заповнити таблицю телефонів його
            // Знайомих, співробітників, родичів, коханих і т. д. Якщо клацнути
            // На кнопці Запис дана таблиця записується на диск у файл у форматі
            // XML. Для спрощення тексту програми передбачений запис в один і той
            // Же файл C: \ tabl.xml. При наступних запусках даної програми таблиця
            // Буде зчитуватися з цього файлу, і користувач може продовжувати
            // Редагування таблиці
            // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
        DataTable^ Таблиця; // Об'явлення объекта "Таблиця"
        DataSet^ НабірДаних; // Об'явлення объекта "НабірДаних"
        private: System::Void Form1_Load(System::Object^ sender, System::EventArgs^ e)
        {
            this->Text = "Простий табличний редактор";
            button1->Text = "Запис";
            Таблиця = gcnew DataTable();
            НабірДаних = gcnew DataSet();
            if (IO::File::Exists("D:\\tabl.xml") == false)
            {
```

```
                // Якщо XML-файлу НЕМАЄ:
                dataGridView1->DataSource = Таблиця;
                // Заповнення "шапки" таблиці
                Таблиця->Columns->Add("Імена");
                Таблиця->Columns->Add("Номери телефонів");
                // Додати об'єкт Таблиця в DataSet
                НабірДаних->Tables->Add(Таблиця);
            }
            else // Якщо XML-файл Є:
            {
                НабірДаних->ReadXml("D:\\tabl.xml");
                // Вміст DataSet у вигляді рядка XML для відладки:
                String ^ РядокXML = НабірДаних->GetXml();
                dataGridView1->DataMember = "Назва таблиці";
                dataGridView1->DataSource = НабірДаних;
            }
        }
        private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)
        {
            // Зберегти файл tabl.xml:
            Таблиця->TableName = "Назва таблиці";
            НабірДаних->WriteXml("D:\\tabl.xml");
        }
    };
}
```

Лістинг 4.2

```
// Програма ілюстрації створення діаграм за табличними даними

#pragma once

namespace WindowsForms {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Сводка для MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
		}

	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::DataVisualization::Charting::Chart^  chart1;
	protected:
	private: System::Windows::Forms::DataGridView^  dataGridView1;
	private: System::Windows::Forms::ComboBox^  comboBox1;

	private:
		/// <summary>
		/// Обязательная переменная конструктора.
		/// </summary>
		System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
		/// <summary>
		/// Требуемый метод для поддержки конструктора — не изменяйте
		/// содержимое этого метода с помощью редактора кода.
		/// </summary>
		void InitializeComponent(void)
		{
			System::Windows::Forms::DataVisualization::Charting::ChartArea^
chartArea1 = (gcnew System::Windows::Forms::DataVisualization::Charting::ChartArea());
```

```
                    System::Windows::Forms::DataVisualization::Charting::Legend^  legend1 =
(gcnew System::Windows::Forms::DataVisualization::Charting::Legend());
                    System::Windows::Forms::DataVisualization::Charting::Series^  series1 =
(gcnew System::Windows::Forms::DataVisualization::Charting::Series());
                    this->chart1 = (gcnew
System::Windows::Forms::DataVisualization::Charting::Chart());
                    this->dataGridView1 = (gcnew System::Windows::Forms::DataGridView());
                    this->comboBox1 = (gcnew System::Windows::Forms::ComboBox());
                    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>chart1))->BeginInit();
                    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>dataGridView1))->BeginInit();
                    this->SuspendLayout();
                    //
                    // chart1
                    //
                    chartArea1->Name = L"ChartArea1";
                    this->chart1->ChartAreas->Add(chartArea1);
                    legend1->Name = L"Legend1";
                    this->chart1->Legends->Add(legend1);
                    this->chart1->Location = System::Drawing::Point(0, 0);
                    this->chart1->Name = L"chart1";
                    series1->ChartArea = L"ChartArea1";
                    series1->Legend = L"Legend1";
                    series1->Name = L"Series1";
                    this->chart1->Series->Add(series1);
                    this->chart1->Size = System::Drawing::Size(477, 300);
                    this->chart1->TabIndex = 0;
                    this->chart1->Text = L"chart1";
                    //
                    // dataGridView1
                    //
                    this->dataGridView1->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;
                    this->dataGridView1->Location = System::Drawing::Point(0, 297);
                    this->dataGridView1->Name = L"dataGridView1";
                    this->dataGridView1->Size = System::Drawing::Size(396, 150);
                    this->dataGridView1->TabIndex = 1;
                    //
                    // comboBox1
                    //
                    this->comboBox1->FormattingEnabled = true;
                    this->comboBox1->Location = System::Drawing::Point(391, 41);
                    this->comboBox1->Name = L"comboBox1";
                    this->comboBox1->Size = System::Drawing::Size(121, 21);
                    this->comboBox1->TabIndex = 2;
                    this->comboBox1->SelectedIndexChanged += gcnew
System::EventHandler(this, &MyForm::comboBox1_SelectedIndexChanged);
                    //
                    // MyForm
                    //
                    this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                    this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                    this->ClientSize = System::Drawing::Size(512, 447);
                    this->Controls->Add(this->comboBox1);
                    this->Controls->Add(this->dataGridView1);
                    this->Controls->Add(this->chart1);
                    this->Name = L"MyForm";
                    this->Text = L"MyForm";
                    this->Load += gcnew System::EventHandler(this, &MyForm::MyForm_Load);
```

```
                    this->Resize += gcnew System::EventHandler(this,
&MyForm::MyForm_Resize);
                    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>chart1))->EndInit();
                    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>dataGridView1))->EndInit();
                    this->ResumeLayout(false);

                }
#pragma endregion
                // Програма використовує елементи керування Chart і DataGridView, виводить
                // графік (діаграму) залежнсті обсягів продажів від часу за місяцями.
                // При цьому в якості джерела даних вказуємо об'єкт класу DataTable
                // Стиль діаграми можна міняти за допомогою переліку з comboBox1
        private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
                this->Text = "Побудова діаграми";
                DataTable ^ Таблиця = gcnew DataTable();
                // У цій таблиці замовляємо дві колонки "Місяць" і "Обсяг продажів":
                Таблиця->Columns->Add("Місяць", String::typeid);
                // У C #: Табліца.Columns.Add("Місяць", typeof(String));
                // Значення в другій колонці призначаємо типу long:
                Таблиця->Columns->Add("Обсяг продажів", long::typeid);
                // У C #: Табліца.Columns.Add("Обсяг продажів", typeof(long));
                // Заповнити перший рядок (ряд) в таблиці:
                DataRow ^ Ряд = Таблиця->NewRow();
                Ряд["Місяць"] = "Травень"; Ряд["Обсяг продажів"] = 15;
                Таблиця->Rows->Add(Ряд);
                // Додаємо другий рядок в таблиці:
                Ряд = Таблиця->NewRow();
                Ряд["Місяць"] = "Червень"; Ряд["Обсяг продажів"] = 35;
                Таблиця->Rows->Add(Ряд);
                // Додаємо третій рядок:
                Ряд = Таблиця->NewRow();
                Ряд["Місяць"] = "Липень"; Ряд["Обсяг продажів"] = 65;
                Таблиця->Rows->Add(Ряд);
                // Додаємо четвертий рядок:
                Ряд = Таблиця->NewRow();
                Ряд["Місяць"] = "Серпень"; Ряд["Обсяг продажів"] = 85;
                Таблиця->Rows->Add(Ряд);
                // Додаємо п'ятий рядок:
                Ряд = Таблиця->NewRow();
                Ряд["Місяць"] = "Вересень"; Ряд["Обсяг продажів"] = 71;
                Таблиця->Rows->Add(Ряд);

                // Наведені нижче властивості можна задати у вікні Властивості для об'єкту
chart1
                // Складену таблицю вказуємо як джерело даних:
                chart1->DataSource = Таблиця;
                // На одному графіку можна зобразити кілька залежностей.
                // Наприклад, перша залежність - обсяги продажів за вказаними
                // Місяців в 2014 році, і друга залежність - продажу по
                // тим же місяцям в 2015 році.
                // В даному графіку ми покажемо тільки одну залежність, дані
                // для відображення цієї залежності назвемо "Series1"
                // На горизонтальній осі відкладаємо назви місяців:
                chart1->Series["Series1"]->XValueMember = "Місяць";
                // А по вертикальній осі відкладаємо обсяги продажів:
                chart1->Series["Series1"]->YValueMembers = "Обсяг продажів";
                // Назва графіка (діаграми):
                chart1->Titles->Add("Обсяг продажів за місяцями");
                // Задаємо тип діаграми - стовпчикова гістограма:
```

```
            chart1->Series["Series1"]->ChartType = System::Windows::Forms::
                  DataVisualization::Charting::SeriesChartType::Column;
            // Тип діаграми може бути іншим, наприклад: Pie, Line і ін.
            chart1->Series["Series1"]->Color = Color::Aqua;
            // Легенду на графіку не відображаємо:
            chart1->Series["Series1"]->IsVisibleInLegend = false;
            // Прив'язка графіка до джерела даних:
            chart1->DataBind();
            // Для сітки даних вказати джерело даних
            dataGridView1->DataSource = Таблиця;

            // Задаємо перелік типів діаграм у об'єкті comboBox1
            // Це можна задати у вікні властивостей comboBox1 в полі Items
            comboBox1->Items->Add("Точкова");              //Point
            comboBox1->Items->Add("Графік");               //Line
            comboBox1->Items->Add("Згладжений графік"); //Spline
            comboBox1->Items->Add("Секторна");                    //Pie
            comboBox1->Items->Add("Пелюсткова");           //Radar
            comboBox1->Items->Add("Гістограма");           //Bar
            comboBox1->Items->Add("Стовпчаста");           //Column
            comboBox1->Text = "Оберіть тип діаграми";

            // Встановлюємо вирівнювання об'єктів в межах форми
            chart1->Dock = System::Windows::Forms::DockStyle::Top;
            dataGridView1->Dock = System::Windows::Forms::DockStyle::Bottom;
            comboBox1->Left = 0;
            comboBox1->Top = 0;
      }

private: System::Void comboBox1_SelectedIndexChanged(System::Object^  sender,
System::EventArgs^  e) {
      // Змінюємо тип діаграми в залежності від обраного рядка в об'єкті  comboBox1
      switch (comboBox1->SelectedIndex)
            {
            case 0:
                  chart1->Series["Series1"]->ChartType =
System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Point;
                  break;
            case 1:
                  chart1->Series["Series1"]->ChartType =
System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Line;
                  break;
            case 2:
                  chart1->Series["Series1"]->ChartType =
System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Spline;
                  break;
            case 3:
                  chart1->Series["Series1"]->ChartType =
System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Pie;
                  break;
            case 4:
                  chart1->Series["Series1"]->ChartType =
System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Radar;
                  break;
            case 5:
                  chart1->Series["Series1"]->ChartType =
System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Bar;
                  break;
            default:
                  chart1->Series["Series1"]->ChartType =
System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Column;
```

```
                    break;
                }
            }
private: System::Void MyForm_Resize(System::Object^  sender, System::EventArgs^  e) {
        // Підбір висоти об'єкту chart1 при зміні розміру вікна
        chart1->Height = this->Height - dataGridView1->Height - 40;
}
};
}
```

Завдання до комп'ютерного практикуму №4

Варіанти

1. Обчислити та запам'ятати значення $z_i = F(x_i, y_i)$ в n різних точках (x_i, y_i), а потім знайти суму всіх отриманих значень функції. Прийняти $n = 14$; $x_{i+1} = x_i + \delta x$ ($x_0 = 2{,}8$; $\delta x = 2{,}1$); $y_{i+1} = y_i + \delta y$ ($y_0 = 1{,}7$; $\delta y = 0{,}3$).

$$F(x_i, y_i) = \left[\sin\left(x_i^2\right) - \cos\left(y_i^2\right)\right] \frac{\sqrt[3]{2x_i - y_i}}{\cos(x_i)}.$$

Побудувати залежність $F = f(x_i)$.

2. Обчислити значення:

$$y_{ij} = z_i \cdot \cos^2 a_j + \sqrt{z_i^3 + a_j \cdot z_i}; \; z_i = \cos(x_i + 1).$$

Прийняти $x_i = 0{,}3; 0{,}8; 1{,}2$; $a_j = 1{,}1; 1{,}5; 1{,}9; 2{,}2$.
Побудувати залежність $y_{ij} = f(a_j)$.

3. Обчислити значення:

$$y_{ij} = z_i^2 + \sqrt{z_i^2 - a_j}; \quad \text{где } z_i = e^{1/x_i} + x_i^2.$$

Прийняти $x_i = 1{,}2; 1{,}8; 1{,}6; 1{,}4$; $a_j = 0{,}4; 0{,}5; 0{,}8$.
Побудувати залежність $y_{ij} = f(x_i)$.

4. Обчислити та запам'ятати значення $z_i = F(x_i, y_i)$ в n різних точках (x_i, y_i), де $n = 14$; $x_{i+1} = x_i + \delta x$ $(x_0 = 1{,}2;\ \delta x = 0{,}25)$; $y_i = x_i^2$.

$$F(x_i, y_i) = \lg^2(x_i \cdot y_i) + \frac{\sqrt{x_i^2 + y_i^2}}{x_i + y_i}.$$

Побудувати залежність $F = f(y_i)$.

5. Обчислити та запам'ятати значення $z_i = F(x_i, y_i)$ в n різних точках (x_i, y_i), де $n = 15$; $x_{i+1} = x_i + \delta x$ ($x_0 = 4{,}2$; $\delta x = 1{,}8$); $y_{i+1} = y_i + \delta y$ ($y_0 = 1{,}82$; $\delta y = 0{,}33$).

$$F(x_i, y_i) = \left[\ln\left(x_i^2\right) - \ln\left(y_i^2\right)\right] \frac{\sqrt[3]{x_i - y_i}}{\operatorname{ctg}(x_i)};$$

Побудувати залежність $F = f(x_i)$.

6. Обчислити та запам'ятати значення $z_i = F(x_i, y_i)$ в n різних точках (x_i, y_i), де $n = 16$; x_i та y_i – довільні додатні числа.

$$F(x_i, y_i) = \sin^2(x_i^2 \cdot y_i) + \cos^3(x_i \cdot y_i^3) - \sqrt{\lg^3 x_i^2}.$$

Побудувати залежність $z_i = f(x_i)$.

7. Обчислити та запам'ятати значення $z_i = F(x_i, y_i)$ в n різних точках (x_i, y_i), де $n = 16$; x_i –

довільні додатні числа; $y_i = \lg^2 x_i^3 + (i-1)/10$.

$$F(x_i, y_i) = \operatorname{tg}^2(x_i - y_i) \cdot (x_i^{\,2} + 0{,}5 \cdot y_i).$$

Побудувати залежність $z_i = f(y_i)$.

8. Обчислити значення:

$$y_{ij} = z_i + \sqrt{z_i^2 + a_j^3 \cdot z_i^{0,3}}; \qquad z_i = e^{-5 \cdot x_i}.$$

Прийняти x_i = 0,1; 0,4; 0,8; 1,2; a_j = 0,2; 0,5; 0,9.
Побудувати залежність $y_{ij} = f(x_i)$.

9. Обчислити значення:

$$y_{ij} = (z_i^{\,2} + 0{,}33)/\operatorname{tg} z_i - b \cdot \sqrt{|z_i^{\,2} - a_j|}; \; z_i = e^{-1/x_i} + x_i^{\,2}.$$

Прийняти x_i = 1,3; 1,5; 1,7; 1,9; a_j = 0,33; 1,12; 3,72; 5,16; b=3,68.
Побудувати залежність $y_{ij} = f(a_j)$.

10. Обчислити значення:

$$y_{ij} = z_i^3 - (z_i + a_j \cdot b)/(a_j^{\,0,4} + z_i^2); \; z_i = e^{-1/x_i} + x_i \cdot b \cdot \cos x_i.$$

Потім знайти суму всіх отриманих значень функції y_{ij}.
Прийняти x_i = 1,3; 1,5; 1,8; 2,0; a_j = 0,7; 0,85; 0,9; b = 5,75.
Побудувати залежність $y_{ij} = f(x_i)$.

11. Обчислити та запам'ятати значення $z_i = F(x_i, y_i)$ в *n* різних точках (x_i, y_i), а потім знайти добуток всіх отриманих значень функції.

$$F(x_i, y_i) = \lg^2(x_i \cdot y_i) + \frac{\sqrt{x_i^{\,2} + y_i^{\,2}}}{x_i + y_i}.$$

Прийняти *n* = 16; $x_{i+1} = x_i + \delta x$ (x_0 = 1,3; δx = 0,2); $y_i = x_i^2$.
Побудувати залежність $z_i = f(x_i)$.

12. Обчислити та запам'ятати значення $z_i = F(x_i, y_i)$ в *n* різних точках (x_i, y_i):

$$F(x_i, y_i) = e^{-1/x} + x^y - y^x,$$

де *n* = 15; $x_{i+1} = x_i + \delta x$ (x_0 = 1,2; δx = 0,3); y_i – довільні додатні числа.
Побудувати залежність $z_i = f(y_i)$.

13. Обчислити та запам'ятати значення:

$$y_{ij} = A_i^2 + A_i \cdot B_i \cdot C_j - B_i^2 \cdot \sqrt{A_i + B_i}; \qquad A_i = \sin^2 x_i; \qquad B_i = e^{1/x_i} + 3,$$

а потім знайти суму всіх отриманих значень функції y_{ij}.
Прийняти x_i = 1,8; 1,4; 1,1; 1,5; C_j = 3,11; 2,21; 2,87.
Побудувати залежність $y_{ij} = f(x_i)$.

14. Обчислити та запам'ятати значення $z_i = F(x_i, y_i)$ в *n* різних точках (x_i, y_i), а потім знайти суму перших п'яти значень та добуток двох останніх.

$$F(x_i, y_i) = e^{-2/x} + 0{,}8 \cdot x^y - 1{,}7 \cdot y^x.$$

Прийняти n = 18; $x_{i+1} = x_i + \delta x$ (x_0 = 1,2; δx=0,55); y_i – довільні додатні числа.
Побудувати залежність $z_i = f(x_i)$.

15. Обчислити значення:

$$y_i = a_i^{b_i} + b_i^{a_i}; \qquad a_i = e^{-1/x_i} + x_i \cdot C; \qquad b_i = \cos^2 x_i + x_i^{0,3}.$$

Прийняти x_i = 0,5; 0,8; 0,9; 1,2; 1,3; 1,5; 1,7; 1,8; 1,9; 2,1; 2,3; 2,4; C=1,86.
Побудувати залежність $a_i = f(x_i)$.

16. Обчислити та запам'ятати значення $z_i = F(x_i, y_i)$ в n різних точках (x_i, y_i),

$$F(x_i, y_i) = 12 \cdot x_i^{0,65} - (1{,}9 \cdot x_i^2 - 0{,}85 \cdot y_i^{0,8}) \cdot tg^2(x_i) / y_i.$$

де n = 18; x_i – довільні додатні числа; y_i – довільні від'ємні числа.
Побудувати залежність $z_i = f(y_i)$.

17. Обчислити та запам'ятати значення:

$$y_{ij} = (A_i \cdot \sin B_j + B_j \cdot \cos A_i) \cdot (A_i^{0,5} + \sqrt[3]{B_j});$$

$$A_i = \cos^2 x_i + x_i^3; \; B_j = \sin^2 a_j + a_j^{0,5}.$$

Прийняти x_i = 1,2; 0,4; 1,3; 0,8; a_j = 0,6; 0,9; 1,9.
Побудувати залежність $y_{ij} = f(A_i)$.

18. Обчислити значення:

$$y_{ij} = z_i + (a_j \cdot z_i \cdot b + b^{0,3}) / \sqrt{z_i^2 + 1}; \quad z_i = e^{-1/x_i} + x_i \cdot b.$$

Прийняти x_i = 1,2; 1,6; 1,9; 1,95; 2,15; a_j = 0,9; 0,95; 0,99; b=1,68.
Побудувати залежність $y_{ij} = f(x_i)$.

19. Обчислити значення:

$$y_{ij} = z^{0,3} + z_i \cdot a_j; \qquad z_i = x_i^2 + \ln x_i.$$

Прийняти x_i = 1,9; 1,5; 1,4; 1,2; a_j = 0,2; 1,1; 1,4.
Побудувати залежність $y_{ij} = f(z_i)$.

20. Обчислити та запам'ятати значення $z_i = F(x_i, y_i)$ в n різних точках (x_i, y_i), а потім знайти суму перших п'яти значень та добуток двох останніх.

$$F(x_i, y_i) = 0{,}8 \cdot x^y - 1{,}7 \cdot y^x + \ln \frac{x}{y}.$$

Прийняти n = 16; $x_{i+1} = x_i + \delta x$ (x_0 = 1.5; δx=0,65); y_i – довільні додатні числа.
Побудувати залежність $z_i = f(x_i)$.

21. Обчислити та запам'ятати значення $z_i = F(x_i, y_i)$ в n різних точках (x_i, y_i),

$$F(x_i, y_i) = \frac{\sqrt{1 - y_i}}{x_i + \operatorname{tg} x_i} \cdot \left(1 + e^{-x_i \cdot y_i}\right).$$

де n = 16; x = x_i * δx (x = 1,8; δx = 1,2); y_i – довільні додатні числа (y_i < 1).
Побудувати залежність $z_i = f(y_i)$.

22. Обчислити значення:

$$y_{ij} = z_i^3 - (z_i + a_j \cdot b)/\left(a_j^{0,3} + z_i^2\right);\ z_i = e^{-2/x_i} + x_i \cdot b \cdot \cos(x_i).$$

Прийняти x_i = 1,2; 1,6; 1,9; 2,2; a_j = 0,75; 0,8; 0,9; 0,95; 1,15; b = 6,74.
Побудувати залежність $y_{ij} = f(a_j)$.

23. Обчислити та запам'ятати значення z_i = F(x_i, y_i) в n різних точках (x_i, y_i),

$$F(x_i, y_i) = 15^{2x-3y} + \sin^2(x^3 + y^4) + x^2 y^3,$$

де n = 18; $x_{i+1} = x_i + e^{\delta x}$ (x_0 = 2; δx = 0,1); y_{i+1}= dy*(1 – y_i) (y_0 =3; dy= -2).
Побудувати залежність $z_i = f(y_i)$.

24. Обчислити значення:

$$y_{ij} = 0{,}88 \cdot z_i + \sqrt{\alpha \cdot z_i^2 + a_j^{2,5} \cdot z_i^{0,45}};\qquad z_i = \lg(3{,}8 \cdot x_i).$$

Прийняти x_i = 1,1; 1,4; 1,8; 2,2; 2,5; 3,1; a_j = 0,2; 0,5; 0,9.
Побудувати залежність $y_{ij} = f(x_i)$.

25. Обчислити та запам'ятати значення:

$$y_{ij} = A_i^{2,2} + 1{,}5 \cdot A_i \cdot B_i \cdot C_j - B_i^{1,5} \cdot \sqrt{A_i + 2B_i};\qquad A_i = 1{,}8 \cdot \cos^3 x_i;\qquad B_i = e^{2/x_i} + 2{,}6,$$

а потім знайти суму всіх отриманих значень функції y_{ij}.
Прийняти x_i = 2,2; 1,9; 1,7; 1,4; 1,2; C_j = 2,31; 2,86; 3,57.
Побудувати залежність $y_{ij} = f(B_i)$.

26. Обчислити значення:

$$y_i = a_i^{2b_i} + b_i^{3a_i};\qquad a_i = e^{2,5/x_i} + 3x_i \cdot C;\qquad b_i = \sin^2 x_i + x_i^{0,5}.$$

Прийняти x_i = 0,3; 0,7; 0,9; 1,1; 1,2; 1,4; 1,7; 1,8; 1,9; 2,2; 2,5; 2,7; C=2,15.
Побудувати залежність $y_i = f(x_i)$.

27. Обчислити значення:

$$y_{ij} = z_i \cdot \cos^2 a_j + \sqrt{z_i^3 + a_j \cdot z_i};\ z_i = \cos(x_i + 1).$$

Прийняти x_i = 0,3; 0,8; 1,2; a_j = 1,1; 1,5; 1,9; 2,2; 2,3; 2,6.
Побудувати залежність $y_{ij} = f(a_j)$.

28. Обчислити значення:

$$y_{ij} = z_i^2 + \sqrt{z_i^2 - a_j};\quad \text{де } z_i = e^{1/x_i} + x_i^2.$$

Прийняти x_i = 1,2; 1,8; 1,6; 1,4; 1,3; a_j = 0,4; 0,5; 0,8.
Побудувати залежність $y_{ij} = f(x_i)$.

29. Обчислити та запам'ятати значення z_i = F(x_i, y_i) в n різних точках (x_i, y_i), де n >14; x_{i+1} =

$x_i + \delta x$ ($x_0 = 1{,}2$; $\delta x = 0{,}25$); $y_i = x_i^2$.

$$F(x_i, y_i) = \lg^2(x_i \cdot y_i) + \frac{\sqrt{x_i^2 + y_i^2}}{x_i + y_i}.$$

Побудувати залежність $z_i = f(y_i)$.

30. Обчислити та запам'ятати значення $z_i = F(x_i, y_i)$ в n різних точках (x_i, y_i), де $n > 20$; $x_{i+1} = x_i + \delta x$ ($x_0 = 3{,}2$; $\delta x = 1{,}2$); $y_{i+1} = y_i + \delta y$ ($y_0 = 1{,}32$; $\delta y = 0{,}33$).

$$F(x_i, y_i) = \left[\ln\left(x_i^2 - 1\right) - \ln\left(y_i^2 + 1\right)\right] \frac{\sqrt[3]{x_i - 3y_i}}{\operatorname{tg}(x_i + 1)}.$$

Побудувати залежність $z_i = f(x_i)$.

Контрольні питання

1) Для чого використовується об’єкт DataGridView?
2) Для чого використовується об’єкт DataTable?
3) Для чого використовується об’єкт DataSet?
4) Як організувати виведення табличних даних у вікні додатка?
5) Як можна зберегти дані таблиці? В якому форматі вони зберігаються та як їх можна переглянути?
6) Для чого використовуються об’єкт Chart? Наведіть його основні можливості.
7) Наведіть типи діаграм, які можна побудувати за допомогою об’єкту Chart.
8) Яким чином зв’язати табличні дані з об’єктом Chart?
9) Як можна програмно змінювати тип діаграми об’єкту Chart?

Комп'ютерний практикум 5

Редагування графічних даних

Мета: вивчити прийоми роботи з графічними даними. Засвоїти методи створення, збереження та відкриття файлів у графічних форматах. Ознайомитись з роботою об'єкту класу Graphics для роботи з графічною інформацією у проектах Windows Forms у Visual C++. Навчитися створювати об'єкти класів Graphics та PictureBox для відображення графічних даних у вікні програми. Ознайомитись з прийомами побудови графіків з використанням об'єкту Graphics. Вивчити прийоми роботи з об'єктом створення багатосторінкових документів TabControl.

Завдання: створити новий пустий проект CLR, додати в нього форму Windows Forms. Додати необхідні об'єкти з Панелі елементів для створення головного, контекстного меню та панелі інструментів. Додати об'єкт TabControl у форму. Створити три вкладки в об'єкті TabControl. На першу вкладку додати об'єкти потрібних класів для введення необхідних вихідних даних згідно отриманого варіанту завдання. На другу вкладку додати об'єкт DataGridView для виведення результатів розрахунку у вигляді таблиці значень. На третю вкладку додати об'єкт PictureBox для відображення графічної залежності. В об'єкті PictureBox побудувати графік вказаної залежності засобами класу Graphics. Передбачити можливість збереження отриманої табличної залежності у обраний користувачем файл формату XML та відкриття вже існуючих файлів XML. Передбачити можливість збереження отриманої графічної залежності у обраний користувачем графічний файл, а також відкриття довільного графічного файлу у об'єкті PictureBox без зміни масштабу з можливістю прокрутки зображення.

Загальні вимоги.

1) Створити проект Windows Forms.
2) Додати в нього необхідні елементи керування та програмний код для реалізації поставленої задачі згідно отриманого варіанту завдання
3) Програмний код має бути чітко структурований.
4) Імена об'єктів мають нести сенсові навантаження.
5) Програмний код має супроводжуватись коментарями в тексті програми.

Вимоги до виконання.

1) Створити новий пустий проект в Visual C++ з використанням створеного шаблону проектів Windows Forms з ім'ям виду Прізвище_КПР5.
2) Додати у форму об'єкт головного меню класу MenuStrip.

3) Додати у форму об'єкт контекстного меню класу `ContextMenuStrip`.
4) Додати у форму об'єкт панелі інструментів класу `ToolStrip`.
5) Додати у форму об'єкт `TabControl` та створити у ньому дві вкладки.
6) Додати на першу вкладку об'єкту `TabControl` об'єкт `DataGridView` для відображення таблиці даних.
7) Додати на другу вкладку об'єкту `TabControl` об'єкт `PictureBox` для відображення графічних даних.
8) Додати об'єкт `Button` на першу вкладку `TabControl` для побудови графіку на основі табличних даних.
9) Додати два об'єкти `Button` на форму для відкриття та збереження графічних файлів.
10) Додати інші необхідні об'єкти для створення програмного інтерфейсу та запрограмувати їх роботу.
11) Передбачити можливість відкриття графічних файлів у об'єкті `PictureBox` з використанням об'єкту `OpenFileDialog`.
12) Передбачити можливість збереження відкритого графічного файлу у обраному форматі з використанням об'єкту `SaveFileDialog`.
13) Запрограмувати побудову графіку на основі табличних даних з об'єкту `DataGridView` за допомогою об'єкту класу `Graphics` в об'єкті `PictureBox`.
14) Передбачити можливість збереження побудованого графіку у файлі обраного графічного формату з використанням об'єкту `SaveFileDialog`.

Теоретичні відомості

5.1 Найпростіше виведення графічного зображення у форму

Поставимо задачу виведення в форму будь-якого зображення - растрового графічного файлу формату BMP, JPEG, PNG або інших форматів. Для вирішення цього завдання запустимо `Visual Studio` і в вікні `Створення проекту` (`New Project`) виберемо в середовищі `CLR` вузла `Visual C++` додаток шаблону `Windows Forms`. У вікні `Оглядача рішень` (`Solution Explorer`) обираємо заголовочний файл форми `MyForm.h`. На екрані з'явиться пуста форма. Потім натискаємо клавішу `F7` на клавіатурі для переходу на вкладку програмного коду. Працювати з графікою в формі можна по-різному. Розглянемо роботу з графікою через перевизначення методу `OnPaint`.

Метод `OnPaint` є членом класу `MyForm` (або `Form` в більш ранніх версіях компілятора). Цей метод можна побачити, набравши `this->` всередині якої-небудь процедури, в списку методів і властивостей об'єкта `MyForm`. Метод `OnPaint` можна перевизначити, тобто додати до вже існуючих функцій власні. Для цього у вікні програмного коду напишемо:

```
protected: virtual void
```

У списку, який з'явився, виберемо OnPaint. Система сама згенерує необхідні рядки процедури, яка підлягає перевизначенню.

Тепер цей програмний код слід доповнити командами для виведення в форму зображення з растрового файлу.

```
#pragma endregion
	// Найпростіше виведення зображення у форму
	protected: virtual void OnPaint(PaintEventArgs ^ e) override
	{
		MyForm::Text = "Малюнок";
		// Створюємо об'єкт для роботи з зображенням
		Image ^ Малюнок = gcnew Bitmap("D:\\PICTUREs\\Емблема_КПІ_Прозора.tif");
		// Виведення зображення в форму
		e->Graphics->DrawImage(Малюнок, 0, 0, this->Width-20, this->Height-40);
		// X = 0, y = 0 - це координати лівого верхнього кута малюнка в
		// системі координат форми: вісь x - вниз, вісь y – вправо
	}
};
}
```

Зверніть увагу, що при вказуванні шляху до файлів, папки розділяються між собою не одинарною, а *подвійною косою рискою* \\

```
"D:\\PICTUREs\\Емблема_КПІ_Прозора.tif"
```

Як видно з тексту програми, спочатку вказуємо заголовок вікна. Далі створюємо об'єкт Малюнок для роботи з зображенням з зазначенням шляху до файлу малюнка. Потім звертаємося безпосередньо до методу малювання зображення у формі DrawImage, витягуючи графічний об'єкт Graphics з аргументу е процедури OnPaint.

Результат роботи програми показаний на рис. 5.1.

Метод DrawImage в даному випадку має п'ять параметрів і визначений наступним чином

```
public:
	void DrawImage(
		Image^ image,
		float x,
		float y,
		float width,
		float height
			)
```

Перший параметр image методу DrawImage – це безпосередньо зображення; другий і третій параметри (x та y) – це координати лівого верхнього кута малюнка в системі координат форми; четвертий і п’ятий параметр (width та height) – це координати правого нижнього кута малюнка в системі координат

форми. При виклику методу DrawImage з таким набором параметрів малюнок буде масштабований таким чином, щоб повністю поміститися у створений кординатами лівого верхнього та правого нижнього кута прямокутник. Співвідношення сторін зображення при цьому не зберігається. Зображення масштабується так, щоб повністю зайняти прямокутник з вказаними координатами. В наведеному вище прикладі зображення буде займати всю поверхню форми.

Рисунок 5.1 – Вікно програми з масштабованим графічним зображенням

Якщо викликати метод DrawImage з трьома параметрами виду

```
public:
void DrawImage(
         Image^ image,
         float x,
         float y
                  )
```

то код може виглядати наступним чином

```
e->Graphics->DrawImage(Малюнок, 0, 0);
```

Даний варіант виклику метода DrawImage малює заданий об'єкт image в зазначеному місці (вказується верхній лівий кут зображення у координатах форми), використовуючи його вихідний фактичний розмір. Тобто зображення виведеться у власному розмірі без змін (рис. 5.2).

Слід зазначити, що це не єдиний спосіб роботи з графікою. Інший спосіб - це викликати той же метод OnPaint побічно через подію форми OnPaint. Такий спосіб роботи з графікою представлений в наступному прикладі.

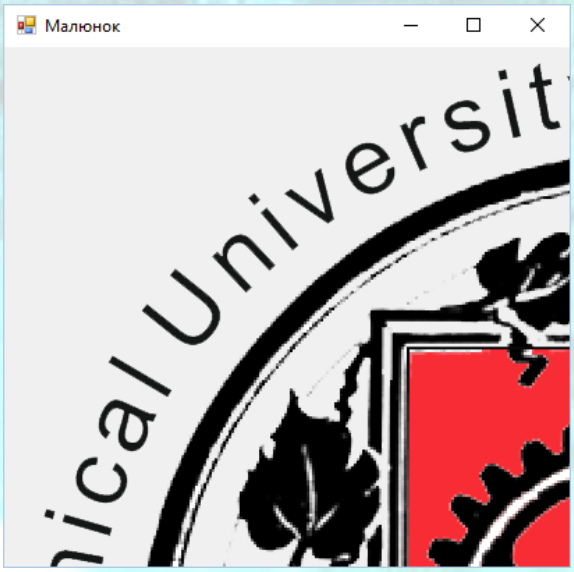

Рисунок 5.2 – Вікно програми з графічним зображенням без масштабування

Приклад програмної реалізації

Приклад виведення графічного зображення з вказаного файлу за допомогою об’єкту класу Image та методу малювання зображення у формі DrawImage графічного об'єкту Graphics з аргументу е процедури OnPaint.

Створюємо процедуру обробки події OnPaint звичайним способом, тобто на вкладці конструктора форми в панелі властивостей вікна Властивості (Properties) клацаємо на значку блискавки і в списку, що з'явився, всіх подій для об'єкта MyForm виберемо подію Paint. Об'єкт Graphics отримуємо з аргументу е події Paint.

```
#pragma endregion
	// Найпростіше виведення зображення в форму
	private: System::Void MyForm_Paint(System::Object^  sender,
		System::Windows::Forms::PaintEventArgs^  e)
	{
		// У властивостях форми клацаємо значок блискавки і в списку, що з'явився
		// всіх подій для об'єкта MyForm виберемо подію Paint.
		// Подія Paint - це подія малювання форми:
		this->Text = "Малюнок"; // Заголовок вікна
		// Створюємо об'єкт для роботи із зображенням:
		Image ^ Малюнок = gcnew Bitmap("D:\\PICTUREs\\ЕмблемаКХТП.png");
		// або Image ^ Малюнок = Image::FromFile("D:\\PICTUREs\\ЕмблемаКХТП.png");
		// Виведення зображення в форму:
		e->Graphics->DrawImage(Малюнок, 5, 5);
	}
};
```

Результат роботи програми зображений на рис. 5.3.

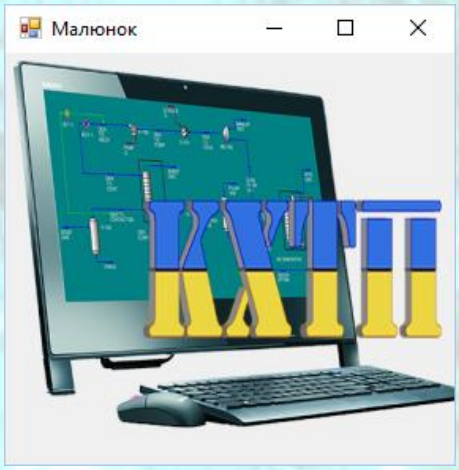

Рисунок 5.3 – Вікно програми з графічним зображенням без масштабування

Приклад виведення зображення з файлу за натисканням кнопки, шляхом створення об’єктів класу Image та Graphics.

Розглянемо ще один спосіб виведення графіки в форму. В цьому випадку при натисканні на командній кнопці відбувається безпосереднє створення об'єкта класу `Graphics`. Програмний код представлений в лістингу нижче.

```
#pragma endregion
	// Найпростіше виведення зображення в форму
	private: System ::
		Void Form1_Load(System::Object ^ sender, System::EventArgs ^ e)
		{
			// Подія завантаження форми:
			MyForm::Text = "Малюнок";
			button1->Text = "Показати малюнок";
		}
	private: System::Void button1_Click(System::Object ^ sender,
		System::EventArgs ^ e)
		{
			// Подія "клацання на кнопці"
			Image ^ Малюнок = gcnew Bitmap("D:\\DOCs\\PICTUREs\\ЕмблемаХТФ.png");
			// Створення графічного об'єкта:
			Graphics ^ Графіка = this->CreateGraphics();
			// Або Graphics ^ Графіка = CreateGraphics ();
			Графіка->DrawImage(Малюнок, 5, 5);
		}
	};
```

Результат роботи програми після натискання кнопки у вікні зображений на рис. 5.4. Повний текст програми наведений в лістингу 5.1.

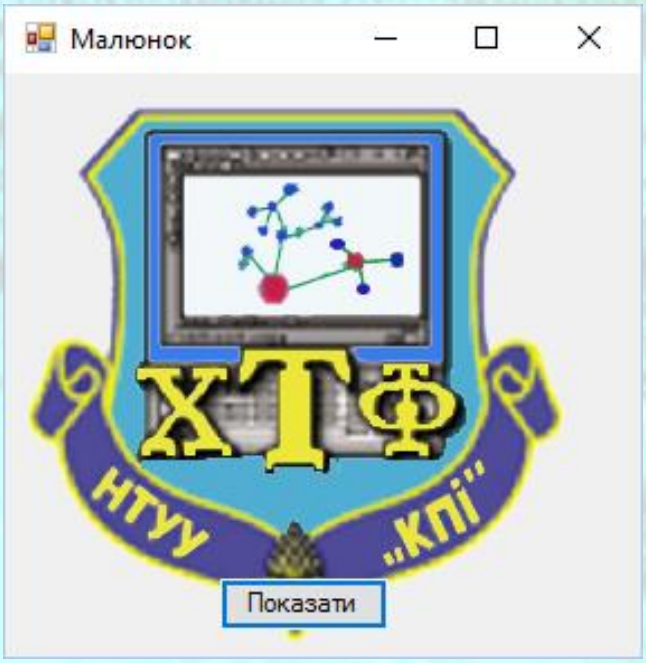

Рисунок 5.4 – Вікно програми з графічним зображенням без масштабування

З розглянутими в даному розділі методами можна працювати не тільки для виведення зображень графічних файлів в форму, але і вирішувати багато інших завдань, пов'язаних з графікою.

5.2 Використання елементу PictureBox для відображення растрового файлу з можливістю прокрутки

Зазвичай для відображення точкових малюнків, малюнків з метафайлів, значків, малюнків з файлів у форматі BMP, JPEG, GIF, PNG і інших використовується об'єкт класу PictureBox (графічне поле). Часто малюнок виявляється занадто великим і не поміщається цілком в межах елемента управління PictureBox. Можна скористатися властивістю елемента SizeMode, вказавши йому значення StretchImage. У цьому випадку зображення буде витягуватися або звужуватися, підлаштовуючись під розмір PictureBox. Дуже часто такий підхід не влаштовує розробника, оскільки відбувається деформація зображення. Вирішення цієї проблеми полягає в організації можливості прокрутки зображення (AutoScroll), але такої властивості у PictureBox немає. Зате така властивість є у елемента управління Panel. Тобто, розмістивши PictureBox на елементі Panel з встановленою властивістю AutoScroll = true і при цьому для PictureBox вказавши SizeMode = AutoSize, ми можемо вирішити задачу прокрутки зображення.

Приклад програмної реалізації

Приклад відкриття довільного графічного файлу з можливістю прокрутки зображення та зберігання його під іншим ім'ям за допомогою об'єктів класу OpenFileDialog та SaveFileDialog.

Запустимо Visual Studio і у вікні Створення проекту (New Project) виберемо в середовищі CLR вузла Visual C++ додаток шаблону Windows Forms. У вікні Оглядача рішень (Solution Explorer) обираємо заголовочний файл форми MyForm.h. На екрані з'явиться пуста форма. З Панелі елементів (Toolbox) перетягнемо на форму елемент управління Panel, а на нього помістимо елемент PictureBox. Розмістимо ще один елемент Panel у формі та додамо всередину об'єкту два елементи Button. Далі перейдемо на вкладку програмного коду за допомогою клавіши F7 і введемо текст, представлений у лістингу.

```
#pragma endregion
	private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e) {
		OpenFileDialog ^ openFileDialog1 = gcnew OpenFileDialog();
		if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)
		{
			pictureBox1->Image = Image::FromFile(openFileDialog1->FileName);
			// Примусово викликаємо обробник події Resize форми
			//для встановлення розмірів об'єктів panel1 та button1
			MyForm_Resize(nullptr, nullptr);
		}

	}

	private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
		this->Text = "Графічний файл";
		button1->Text = "Відкрити";
		button2->Text = "Зберегти";
		panel2->Height = 40;
		panel2->Dock = System::Windows::Forms::DockStyle::Bottom;
		button1->Dock = System::Windows::Forms::DockStyle::Right;
		panel1->Dock = System::Windows::Forms::DockStyle::Top;
		pictureBox1->SizeMode = PictureBoxSizeMode::AutoSize;
		panel1->AutoScroll = true; // Дозволяємо прокрутку зображення
		button2->Dock = System::Windows::Forms::DockStyle::Fill;
	}

	private: System::Void MyForm_Resize(System::Object^  sender, System::EventArgs^  e) {
		// Встановлення розмірів об'єктів panel1 та button1
		panel1->Height = this->Height - panel2->Height - 40;
		button1->Width = this->panel2->Width / 2;
	}

	private: System::Void button2_Click(System::Object^  sender, System::EventArgs^  e) {
		// Відобразити вікно SaveFileDialog щоб користувач міг зберегти зображення
		SaveFileDialog ^ saveFileDialog1 = gcnew SaveFileDialog();
		saveFileDialog1->Filter =
			"Зображення JPeg|*.jpg|Зображення Bitmap|*.bmp|Зображення Gif|*.gif";
		saveFileDialog1->Title = "Зберегти файл зображення";
		saveFileDialog1->ShowDialog();
```

```
                // Якщо ім'я файлу не пустий рядок, зберегти зображення
                if (saveFileDialog1->FileName != "")
                {
                        // Зберегти зображення через FileStream, створений методом OpenFile
                        System::IO::FileStream ^ fs =
                                safe_cast<System::IO::FileStream^>(saveFileDialog1->OpenFile());
                        // Можна зберегти зображення у відповідному форматі ImageFormat на
                        // основі типу файлу, обраного в діалоговому вікні.
                        // Тип файлу береться з властивості FilterIndex
                        switch (saveFileDialog1->FilterIndex)
                        {
                        case 1:
                                this->pictureBox1->Image->Save(fs,
                                        System::Drawing::Imaging::ImageFormat::Jpeg);
                                break;
                        case 2:
                                this->pictureBox1->Image->Save(fs,
                                        System::Drawing::Imaging::ImageFormat::Bmp);
                                break;
                        case 3:
                                this->pictureBox1->Image->Save(fs,
                                        System::Drawing::Imaging::ImageFormat::Gif);
                                break;
                        }
                        fs->Close();
                }
}
};
}
```

Окрім відображення графічного файлу з можливістю прокрутки, дана програма дозволяє вибирати довільний файл для відкриття за допомогою елементу діалогового вікна `OpenFileDialog`. Відкритий файл можна зберегти під іншим ім'ям і в іншому графічному форматі за допомогою елементу діалогового вікна `SaveFileDialog`.

Об'єкти класів `OpenFileDialog` та `SaveFileDialog` створюються програмно, хоча обидва цих елементи можна було перетягти з `Панелі елементів` у форму вручну і тоді б наступні рядки коду не потрібно було б писати:

```
OpenFileDialog ^ openFileDialog1 = gcnew OpenFileDialog();
SaveFileDialog ^ saveFileDialog1 = gcnew SaveFileDialog();
```

Також програма виконує масштабування елементів свого інтерфейсу при зміні розміру вікна (рис. 5.5).

Всі властивості обробника події `MyForm_Load` можна було задати в режимі проектування у вікні `Властивості` для кожного з наведених там об'єктів і тоді даний обробник події програмувати було б непотрібно.

Обробник події `MyForm_Resize`, розом з встановленими властивостями розмірів та розташування елементів інтерфейсу у вікні, відповідає за правильне масштабування інтерфейсу програми при зміні розміру вікна.

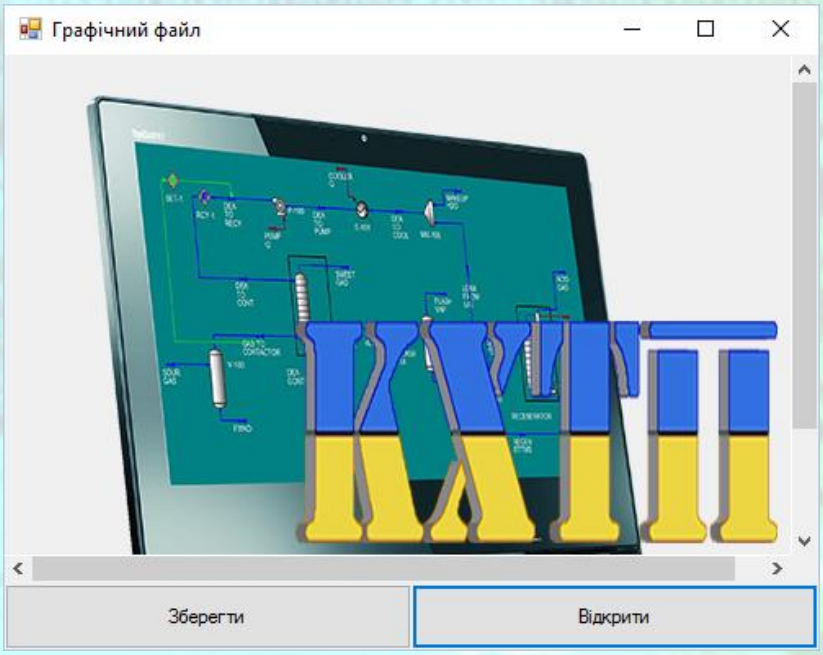

Рисунок 5.5 – Вікно програми з графічним зображенням

Обробник події `button1_Click` виникає при натисканні на кнопку Відкрити у вікні програми та відповідає за відображення діалогового вікна відкриття файлу і розміщення обраного зображення в об'єкті `pictureBox1`.

Обробник події `button2_Click` виникає при натисканні на кнопку Зберегти у вікні програми та дозволяє за допомогою діалогового вікна збереження файлу зберегти відкритий файл під іншим ім'ям в одному з трьох графічних форматів, які задаються у властивості `Filter` об'єкту `saveFileDialog1`. Повний текст програми наведений в лістингу 5.2.

5.3 Побудова графіку методами класу Graphics

Графічне зображення можна створювати в ході виконання програми шляхом пубудови екремих його елементів. У наступному прикладі ми, використовуючи в якості вихідних даних, наприклад, обсяги продажів будь-яких товарів по місяцях, побудуємо графік за точками. Зрозуміло, що таким же чином можна побудувати будь-який графік для інших прикладних цілей.

Приклад програмної реалізації

Приклад побудови графіку в ході виконання програми за допомогою об'єктів класів Bitmap, Graphics і Pen та методів DrawString, DrawLine та DrawEllipse класу Graphics.

Для вирішення цього завдання запустимо Visual Studio і в вікні Створення проекту (New Project) виберемо в середовищі CLR вузла Visual C++ додаток шаблону Windows Forms. Перенесемо з Панелі елементів

(Toolbox) в проектовану форму елемент управління PictureBox (графічними поле) і об'єкт контейнер Panel. Властивість Dock об'єкта Panel встановіть у значення Bottom у вікні Властивості (Properties) для того щоб даний об'єкт розтягувався на всю ширину вікна вздовж нижньої його частини. Властивість Dock об'єкта PictureBox встановіть у значення Fill у вікні Властивості (Properties) для того щоб даний об'єкт зайняв всю вільну область вікна окрім тієї, що займає об'єкт Panel.

В середині об'єкту Panel розмістіть командну кнопку Button та встановіть її властивість Dock у значення Fill для того, щоб кнопка займала всю вільну область об'єкта Panel. Далі перейдемо на вкладку програмного коду і введемо текст, представлений нижче.

```
#pragma endregion
        // Програма малює графік продажів по місяцях. Зрозуміло, що таким же чином
        // Можна побудувати будь-який графік по точках для інших прикладних цілей
        // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
        // Вихідні дані для побудови графіка (тобто вихідні точки):
        array <String ^> ^ Months;
        array <int> ^ Sales;
        Graphics ^ Графіка;
        // Далі, створюємо об'єкт Bitmap, який має
        // Той же розмір і роздільну здатність, що і PictureBox
        Bitmap ^ Растр;
        int ВідступЛіворуч, ВідступПраворуч, ВідступЗнизу, ВідступЗверху;
        int ДовжинаВертОсі, ДовжинаГоризОсі, YГоризОсі, XВертОсі, Xmax, XПочЕпюри;
        // Крок градування по горизонтальній і вертикальній осях:
        double ГоризКрок;
        int ВертКрок;
        // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

private: System::Void button1_Click(System::Object ^ sender, System::EventArgs ^ e)
{// Побудова графіку:
        Months = gcnew array <String ^> { "Січ", "Лют", "Бер", "Квіт", "Трав",
                                "Черв", "Лип", "Серп", "Вер", "Жовт", "Лист", "Груд"};
        Sales = gcnew array <int> {335, 414, 572, 629, 750, 931, 753, 599, 422,
                                    301, 245, 155};
        ВідступЛіворуч = 35; ВідступПраворуч = 15;
        ВідступЗнизу = 20; ВідступЗверху = 10;
        Растр = gcnew Bitmap(pictureBox1->Width, pictureBox1->Height,
                                pictureBox1->CreateGraphics());
        pictureBox1->BorderStyle = BorderStyle::FixedSingle;//Рамка навколо графіка
        YГоризОсі = pictureBox1->Height - ВідступЗнизу;
        XВертОсі = pictureBox1->Width - ВідступЛіворуч;
        Xmax = pictureBox1->Width - ВідступПраворуч;
        ДовжинаГоризОсі = pictureBox1->Width - (ВідступЛіворуч + ВідступПраворуч);
        ДовжинаВертОсі = YГоризОсі - ВідступЗверху;
        ГоризКрок = (double)(ДовжинаГоризОсі / Sales->Length + 3);
        ВертКрок = (int)(ДовжинаВертОсі / 10);
        XПочЕпюри = ВідступЛіворуч;//Початкове значення графіку по вісі X
        // Послідовно викликаємо наступні процедури:
        Графіка = Graphics::FromImage(Растр);
        МалюємоВісі();
        МалюємоГоризЛінії();
        МалюємоВертЛінії();
        МалюємоЕпюру();
```

```
        pictureBox1->Image = Растр;
        // Звільняємо ресурси, використовувані об'єктом класу Graphics:
        delete Графіка; // - Еквівалент C#: Графіка.Dispose
} // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

private: void МалюємоВісі()
{
        Pen ^ Перо = gcnew Pen(Color::Black, 2);
        // Малювання вертикальної осі координат:
        Графіка->DrawLine(Перо, ВідступЛіворуч, YГоризОсі, ВідступЛіворуч, ВідступЗверху);
        // Малювання горизонтальної осі координат:
        Графіка->DrawLine(Перо, ВідступЛіворуч, YГоризОсі, Xmax, YГоризОсі);
        Drawing::Font ^ Шрифт = gcnew Drawing::Font("Arial", 8);
        // Підписуємо значення по вісі Y
        for (int i = 1; i <= 10; i++)
        {// Малюємо "вусики" на вертикальній координатної осі:
                int Y = YГоризОсі - i * ВертКрок;
                Графіка->DrawLine(Перо, ВідступЛіворуч - 5, Y, ВідступЛіворуч, Y);
                // Підписуємо значення продажів через кожні 100 одиниць:
                Графіка->DrawString((i * 100).ToString(), Шрифт, Brushes::Black, 2, Y - 5.F);
        }
        // Підписуємо значення по вісі X
        for (int i = 0; i <= Months->Length - 1; i++)
        {// Малюємо "вусики" на горизонтальній координатної осі:
                int X = ХПочЕпюри + (int)(ГоризКрок * (i + 1));
                Графіка->DrawLine(Перо, X, YГоризОсі + 5, X, YГоризОсі );
                // Підписуємо місяці на горизонтальній осі:
                Графіка->DrawString(Months[i], Шрифт, Brushes::Black, ВідступЛіворуч
                                        - 10.F + (int)(i * ГоризКрок), YГоризОсі + 4.F);
        }
} // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

private: void МалюємоГоризЛінії()
{
        Pen ^ ТонкеПеро = gcnew Pen(Color::LightGray, 1);
        for (int i = 1; i <= 10; i++)
        {// Малюємо горизонтальні майже "прозорі" лінії:
                int Y = YГоризОсі - ВертКрок * i;
                Графіка->DrawLine(ТонкеПеро, ВідступЛіворуч, Y, Xmax, Y);
        }
        delete ТонкеПеро; // - Еквівалент C#: ТонкоеПеро.Dispose();
} // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

private: void МалюємоВертЛінії()
{// Малюємо вертикальні майже "прозорі" лінії
        Pen ^ ТонкеПеро = gcnew Pen(Color::Bisque, 1);
        for (int i = 0; i <= Months->Length - 1; i++)
        {
                int X = ХПочЕпюри + (int)(ГоризКрок * i);
                Графіка->DrawLine(ТонкеПеро, X, ВідступЗверху, X, YГоризОсі - 4);
        }
        delete ТонкеПеро;
} // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

private: void МалюємоЕпюру()
{
        double ВертМасштаб = (double)ДовжинаВертОсі / 1000;
        // Значення ординат на екрані:
        array <int> ^ Y = gcnew array <int>(Sales->Length);
        // Значення абсцис на екрані:
        array <int> ^ X = gcnew array <int>(Sales->Length);
```

```
        for (int i = 0; i <= Sales->Length - 1; i++)
        {// Обчислюємо графічні координати точок:
               Y[i] = YГоризОсі - (int)(Sales[i] * ВертМасштаб);
               // Віднімаємо значення продажів, оскільки вісь Y екрану спрямована вниз
               X[i] = XПочЕпюри + (int)(ГоризКрок * i);
        }
        // Малюємо перше коло:
        Pen ^ Перо = gcnew Pen(Color::Blue, 3);
        Графіка->DrawEllipse(Перо, X[0] - 2, Y[0] - 2, 4, 4);
        for (int i = 0; i <= Sales->Length - 2; i++)
        {// Цикл по лініях між точками:
               Графіка->DrawLine(Перо, X[i], Y[i], X[i + 1], Y[i + 1]);
               // Віднімаємо 2, оскільки діаметр (ширина) точки = 4:
               Графіка->DrawEllipse(Перо, X[i + 1] - 2, Y[i + 1] - 2, 4, 4);
        }
} // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e)
{ // Встановлення заголовку вікна та тексту кнопки при запуску вікна програми
        this->Text = "Побудова графіка";
        button1->Text = "Намалювати графік";
}
};
}
```

Після того, як програмний код набраний, потрібно перевірити чи присвоєна подія Click для об'єкту Button1 у вікні Властивості (Properties) на вкладці Events (події). Якщо ні, то потрібно обрати button1_Click зі спадаючого списку у полі події Click. Аналогіно потрібно перевірити подію Load для форми і за необхідності вказати потрібну подію зі списку.

Якщо все зроблено вірно, то після запуску додатку з'явиться пусте вікно з кнопкою. По натисканню кнопки «Намалювати графік» у вікні з'явиться графічна залежність (рис. 5.6).

Якщо розмір графіку замалий, можна розтягти вікно і знову натиснути кнопку «Намалювати графік». Графік буде перебудований у новому масштабі таким чином, щоб зайняти весь вільний простір вікна.

Як видно з тексту програми, спочатку оголошуємо деякі змінні так, щоб вони були видні з усіх процедур класу. Строковий масив Months містить назви місяців, які користувач нашого програмного коду може змінювати в залежності від контексту графіка, який будується. У будь-якому випадку записані рядки в цьому масиві будуть відображатися по горизонтальній осі графіка. Масив цілих чисел Sales містить обсяги продажів по кожному місяцю, вони відповідають ординатам графіка. Обидва масиву повинні мати однакову розмірність, але не обов'язково рівну дванадцяти.

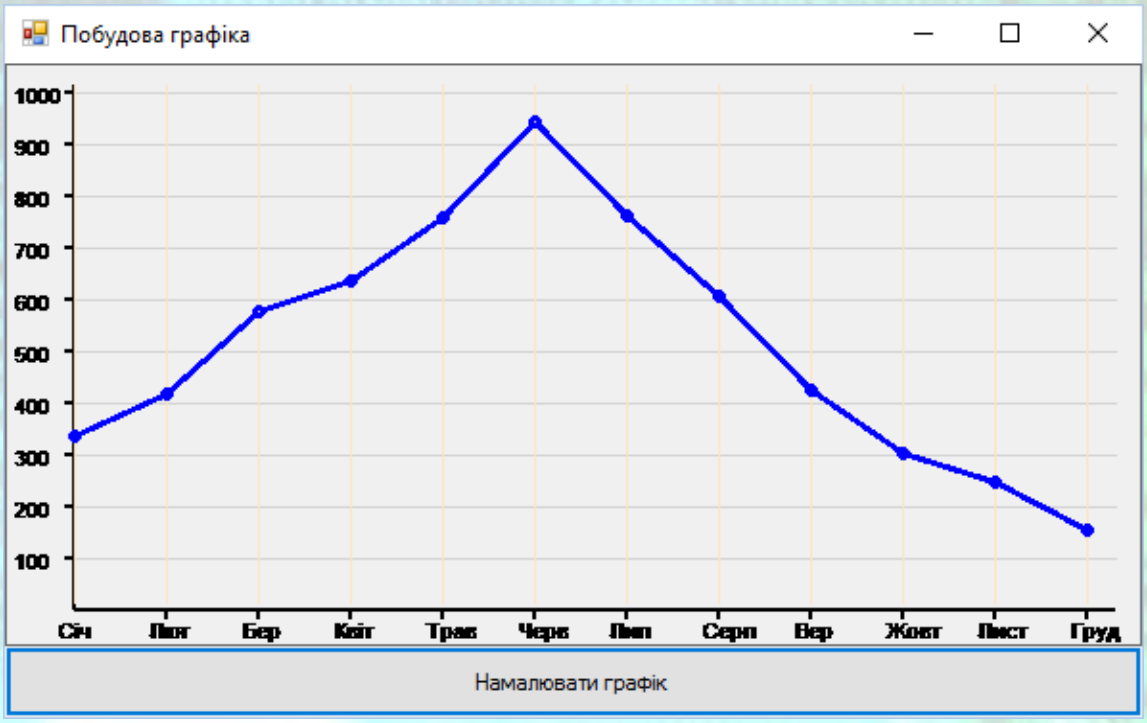

Рисунок 5.6 – Вікно програми побудови графіка

При обробці події натискання кнопки `Button` створюємо об'єкт класу `Graphics`, використовуючи елемент управління `PictureBox` (графічне поле), а потім, викликаючи відповідні процедури, поетапно малюємо координатні осі, сітку з горизонтальних і вертикальних ліній і безпосередньо епюру. Щоб успішно, мінімальними зусиллями і з можливістю подальшого вдосконалення програми побудувати графік, слід якомога зрозуміліше назвати деякі ключові інтервали і координати на малюнку. Назви цих інтервалів повинні бути змістовними. Скажімо, змінна `ВідступЛіворуч` зберігає число пікселів, на яке слід відступити, щоб побудувати на графіку, наприклад, вертикальну вісь продажів. Крім очевидних назв згадаємо змінну `YГоріз0сі`, це графічна ордината (вісь x спрямована зліва направо, а вісь y - зверху вниз) горизонтальної осі графіка, на якій підписуються місяці. Змінна `Xmax` містить в собі значення максимальної абсциси (див. рис. 5.6), правіше якої вже ніяких побудов немає. Змінна `ХПочЕпюри` - це значення абсциси першої побудованої точки графіка. Використовуючі такі зрозумілі з контексту назви змінних, та ще й українською мовою, ми значно полегшуємо весь процес програмування, спрощуємо супровід і модифікацію програми. Текст програми наведений в лістингу 5.5.

5.4 Створення багатосторінкового інтерфейсу

У випаках, коли програмний інтерфейс повинен містити велику кількість графічних елементів, наприклад, полей введення, кнопок, графічних зображень, переліків і т. ін., доцільно розмістити у вікні програми контейнер класу

TabControl. Об’єкт даного класу дозволяє створювати у вікні програми кілька сторінок з закладками.

Об’єкт класу TabControl управляє пов'язаним набором сторінок вкладок. Сторінки-вкладки представлені об’єктами класу TabPage, які додаються за допомогою властивості TabPages об’єкту TabControl. Порядок сторінок вкладок в цій колекції відповідає порядку, в якому вкладки з'являються в елементі управління.

Користувач може змінити поточний порядок об’єктів класу TabPage клацнувши на одній з вкладок в елементі управління. Можна також програмно змінити поточну сторінку TabPage за допомогою однієї з наступних властивостей елементу TabControl: SelectedIndex, SelectedTab.

У .NET Framework 2.0, також можна використовувати один з наступних методів: SelectTab, DeselectTab.

.NET Framework 2.0, дозволяє реагувати на зміну поточної вкладки шляхом обробки однієї з наступних подій: Deselecting, Deselected, Selecting, Selected.

Вкладки в TabControl є частиною TabControl, але не є частиною окремих елементів управління TabPage. Члени класу TabPage, такі як властивість ForeColor, впливають тільки на клієнтську область вкладки, але не на символи заголовку вкладки. Крім того метод Hide об’єкту TabPage сховає лише клієнтську частину вкладки з елементами, які розміщені в ній, але не сховає заголовок вкладки. Щоб прибрати вкладку, необхідно видалити об’єкт TabPage з колекції елементу управління TabControl::TabPages.

Наступні події не викликаються для об’єкту класу TabControl, якщо не існує принаймні один об’єкт TabPage в колекції TabControl::TabPages: Control::Click, Control::DoubleClick, Control::MouseDown, Control::MouseUp, Control::MouseHover, Control::MouseEnter, Control::MouseLeave і Control::MouseMove. Якщо є хоча б один елемент TabPage в колекції, і користувач взаємодіє з заголовком елемента управління вкладки (де відображаються значення властивості TabPage::Text, які за замовчуванням співпадають з іменами об’єктів TabPage), TabControl викликає відповідну подію. Проте якщо користувач працює в межах ClientRectangle (клієнтської області) сторінки вкладки TabPage, викликається відповідна подія для активного об’єкту TabPage.

Елементи управління, що містяться в TabPage не створюються, поки не буде показана сторінка вкладки, і будь-які прив'язки даних в цих елементах керування не активуються, поки не буде показана сторінка вкладки.

Якщо візуальні стилі включені і властивості Alignment присвоєно значення, відмінне від Top, вміст вкладки може відображатися неправильно. Щоб обійти цю проблему, можна намалювати вміст вкладки самостійно.

Коли властивості Alignment присвоєно значення, відмінне від Top і властивості Appearance присвоєно значення, відмінне від Normal, вміст сторінки може відображатися неправильно.

Приклад програмної реалізації

Приклад відкриття різних типів файлів з використанням діалогового вікна OpenFileDialog на різних сторінках TabPage елементу TabControl.

Розглянемо приклад створення програми, яка по натисканню кнопки відкриватиме графічні файли чи файли формату RTF в залежності від того, яка сторінка об'єкту класу TabControl активна.

Для вирішення цього завдання запустимо Visual Studio і в вікні Створення проекту (New Project) виберемо в середовищі CLR вузла Visual C++ додаток шаблону Windows Forms. Перенесемо з Панелі елементів (Toolbox) в проектовану форму елемент управління кнопка Button. Для властивості Dock кнопки у вікні Properties задамо значення Bottom, щоб кнопка розтягувалась вздовж всієї нижньої частини форми. У властивості Text кнопки вкажемо значення «Відкрити файл».

Перетягнемо з розділу Контейнери вікна Панелі елементів об'єкт класу TabControl. Для властивості Dock елементу TabControl у вікні Properties задамо значення Fill, щоб вкладки сторінок розтягувалась по всій вільній області форми.

На першій сторінці в об'єкті tabPage1 класу TabPage розмістимо об'єкт класу PictureBox з розділу Стандартні елементи управління вікна Панелі елементів. Для властивості Dock об'єкту pictureBox1 задамо значення Fill. Для властивості Text об'єкту tabPage1 вкажемо значення «Графічний файл».

На другій сторінці в об'єкті tabPage2 класу TabPage розмістимо об'єкт класу RichTextBox з розділу Стандартні елементи управління вікна Панелі елементів. Для властивості Dock об'єкту richTextBox1 задамо значення Fill. Для властивості Text об'єкту tabPage2 вкажемо значення «Файл RTF».

Надалі в обробник події Click кнопки додамо наступний програмний код.

```
private: System::Void buttonВідкрити_Click(System::Object^  sender, System::EventArgs^  e) {
	if (tabControl1->SelectedIndex == 0) //Перевірка чи активна перша вкладка
	{	//Тоді відкриваємо графічний файл
		//Створюємо об'єкт класу OpenFileDialog
		OpenFileDialog ^ openFileDialog1 = gcnew OpenFileDialog();
		//Встановлюємо в діалоговому вікні фільтр для відкриття файлів
		openFileDialog1->Filter = L"Файли зображень | *.bmp;*.jpg;*.gif; *.png";
		if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK
		 && openFileDialog1->FileName->Length > 0)
		//Якщо користувач обрав в діалоговому вікні файл і його ім'я не пусте
		{
			//Відкриваємо графічний файл в об'єкті pictureBox1
			pictureBox1->Image = Image::FromFile(openFileDialog1->FileName);
			//Розміщуємо ім'я відкритого файлу в заголовку вкладки
			tabPage1->Text = Path::GetFileName(openFileDialog1->FileName);
		}
	}
	else //Активна друга вкладка
	{	//Тоді відкриваємо файл *.rtf
		//Створюємо об'єкт класу OpenFileDialog
		OpenFileDialog ^ openFileDialog1 = gcnew OpenFileDialog();
		//Встановлюємо в діалоговому вікні фільтр для відкриття файлів
		openFileDialog1->Filter = L"Текстові файли | *.rtf";
		if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK
		 && openFileDialog1->FileName->Length > 0)
		//Якщо користувач обрав в діалоговому вікні файл і його ім'я не пусте
		{
			//Відкриваємо файл формату RTF в об'єкті richTextBox1
			richTextBox1->LoadFile(openFileDialog1->FileName);
			//Розміщуємо ім'я відкритого файлу в заголовку вкладки
			tabPage2->Text = Path::GetFileName(openFileDialog1->FileName);
		}
	}
}
```

При натисканні кнопки buttonВідкрити перевіряється яка з вкладок активна

```
	if (tabControl1->SelectedIndex == 0) //Перевірка чи активна перша вкладка
	{	//Тоді відкриваємо графічний файл


	}
	else //Активна друга вкладка
	{	//Тоді відкриваємо файл *.rtf

	}
```

Якщо активна перша вкладка, створюємо об’єкт класу діалогового вікна відкриття файлу `OpenFileDialog` і задаємо фільтр файлів в діалоговому вікні, щоб користувач не міг відкрити файли неприпустимого типу.

```
//Створюємо об'єкт класу OpenFileDialog
OpenFileDialog ^ openFileDialog1 = gcnew OpenFileDialog();
//Встановлюємо в діалоговому вікні фільтр для відкриття файлів
openFileDialog1->Filter = L"Файли зображень | *.bmp;*.jpg;*.gif; *.png";
```

Після цього відкриваємо діалогове вікно і перевіряємо чи обрав користувач файл для відкриття.

```
if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK
 && openFileDialog1->FileName->Length > 0)
//Якщо користувач обрав в діалоговому вікні файл і його ім'я не пусте
```

Якщо дана умова виконується, відкриваємо графічний файл в об'єкті pictureBox1 та записуємо ім'я відкритого файлу у заголовок вкладки.

```
//Відкриваємо графічний файл в об'єкті pictureBox1
pictureBox1->Image = Image::FromFile(openFileDialog1->FileName);
//Розміщуємо ім'я відкритого файлу в заголовку вкладки
tabPage1->Text = Path::GetFileName(openFileDialog1->FileName);
```

Функція GetFileName() класу Path з простору імен System::IO виокремлює ім'я файлу з шляху та імені файлу, які зберігаються у властивості FileName об'єкту openFileDialog1. Для можливості використання даної функції потрібно підключити відповідний простір імен.

```
using namespace System::IO; //Підключення простору імен для методу Path::GetFileName()
```

Аналогічним чином виконується відкриття файлів RTF в об'єкті richTextBox1, якщо активною є друга вкладка tabPage2.

```
//Відкриваємо файл формату RTF в об'єкті richTextBox1
richTextBox1->LoadFile(openFileDialog1->FileName);
//Розміщуємо ім'я відкритого файлу в заголовку вкладки
tabPage2->Text = Path::GetFileName(openFileDialog1->FileName);
```

Результат роботи програми зображений на рис. 5.7.

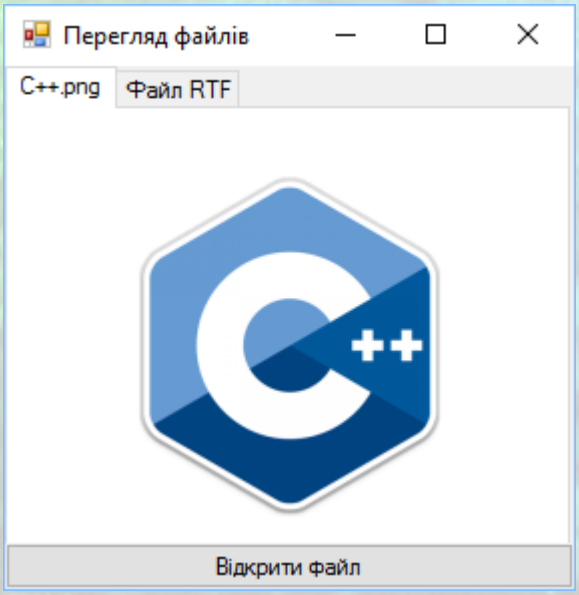

Рисунок 5.7 – Вікно програми відкриття різних типів файлів на двох окремих сторінках

Для того, щоб відкриті зображення масштабувались в межах розмірів вікна, для об'єкту `pictureBox1` було встановлене значення `Zoom` властивості `SizeMode`.

Повний текст програми наведений нижче в лістингу 5.7.

Програмний код

Лістинг 5.1

```
//Програма ілюстрації відкриття графічного файлу у формі

#pragma once

namespace WindowsForms {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Сводка для MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
		}

	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::Button^  button1;
	protected:

	private:
		/// <summary>
		/// Обязательная переменная конструктора.
		/// </summary>
		System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
		/// <summary>
		/// Требуемый метод для поддержки конструктора — не изменяйте
		/// содержимое этого метода с помощью редактора кода.
		/// </summary>
		void InitializeComponent(void)
		{
			this->button1 = (gcnew System::Windows::Forms::Button());
			this->SuspendLayout();
```

```
                    //
                    // button1
                    //
                    this->button1->Location = System::Drawing::Point(96, 226);
                    this->button1->Name = L"button1";
                    this->button1->Size = System::Drawing::Size(75, 23);
                    this->button1->TabIndex = 0;
                    this->button1->Text = L"button1";
                    this->button1->UseVisualStyleBackColor = true;
                    this->button1->Click += gcnew System::EventHandler(this,
&MyForm::button1_Click);
                    //
                    // MyForm
                    //
                    this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                    this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                    this->ClientSize = System::Drawing::Size(284, 261);
                    this->Controls->Add(this->button1);
                    this->Name = L"MyForm";
                    this->Text = L"MyForm";
                    this->Load += gcnew System::EventHandler(this, &MyForm::Form1_Load);
                    this->ResumeLayout(false);

             }
#pragma endregion
      // Найпростіше виведення зображення в форму
      private: System ::
             Void Form1_Load(System::Object ^ sender, System::EventArgs ^ e)
             {
                    // Подія завантаження форми:
                    MyForm::Text = "Малюнок";
                    button1->Text = "Показати малюнок";
             }
      private: System::Void button1_Click(System::Object ^ sender,
             System::EventArgs ^ e)
             {
                    // Подія "клацання на кнопці"
                    Image ^ Малюнок = gcnew Bitmap("D:\\DOCs\\PICTUREs\\ЕмблемаХТФ.png");
                    // Створення графічного об'єкта:
                    Graphics ^ Графіка = this->CreateGraphics();
                    // Або Graphics ^ Графіка = CreateGraphics ();
                    Графіка->DrawImage(Малюнок, 5, 5);
             }
      };
}
```

Лістинг 5.2

```
//Приклад використання діалогових вікон OpenFileDialog та SaveFileDialog
//а також відкриття графічних зображень в об'єкті PictureBox

#pragma once

namespace WindowsForms {
	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Сводка для MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
		}

	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::Panel^  panel1;
	private: System::Windows::Forms::PictureBox^  pictureBox1;

	private: System::Windows::Forms::Panel^  panel2;
	private: System::Windows::Forms::Button^  button1;
	private: System::Windows::Forms::Button^  button2;
	protected:

	private:
		/// <summary>
		/// Обязательная переменная конструктора.
		/// </summary>
		System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
		/// <summary>
		/// Требуемый метод для поддержки конструктора — не изменяйте
		/// содержимое этого метода с помощью редактора кода.
		/// </summary>
		void InitializeComponent(void)
```

```
        {
            this->panel1 = (gcnew System::Windows::Forms::Panel());
            this->pictureBox1 = (gcnew System::Windows::Forms::PictureBox());
            this->panel2 = (gcnew System::Windows::Forms::Panel());
            this->button2 = (gcnew System::Windows::Forms::Button());
            this->button1 = (gcnew System::Windows::Forms::Button());
            this->panel1->SuspendLayout();
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>pictureBox1))->BeginInit();
            this->panel2->SuspendLayout();
            this->SuspendLayout();
            //
            // panel1
            //
            this->panel1->Controls->Add(this->pictureBox1);
            this->panel1->Location = System::Drawing::Point(0, 0);
            this->panel1->Name = L"panel1";
            this->panel1->Size = System::Drawing::Size(239, 127);
            this->panel1->TabIndex = 0;
            //
            // pictureBox1
            //
            this->pictureBox1->Location = System::Drawing::Point(50, 25);
            this->pictureBox1->Name = L"pictureBox1";
            this->pictureBox1->Size = System::Drawing::Size(100, 50);
            this->pictureBox1->TabIndex = 2;
            this->pictureBox1->TabStop = false;
            //
            // panel2
            //
            this->panel2->Controls->Add(this->button2);
            this->panel2->Controls->Add(this->button1);
            this->panel2->Location = System::Drawing::Point(0, 146);
            this->panel2->Name = L"panel2";
            this->panel2->Size = System::Drawing::Size(239, 115);
            this->panel2->TabIndex = 2;
            //
            // button2
            //
            this->button2->Location = System::Drawing::Point(99, 24);
            this->button2->Name = L"button2";
            this->button2->Size = System::Drawing::Size(75, 23);
            this->button2->TabIndex = 3;
            this->button2->Text = L"button2";
            this->button2->UseVisualStyleBackColor = true;
            this->button2->Click += gcnew System::EventHandler(this,
&MyForm::button2_Click);
            //
            // button1
            //
            this->button1->Location = System::Drawing::Point(50, 72);
            this->button1->Name = L"button1";
            this->button1->Size = System::Drawing::Size(138, 31);
            this->button1->TabIndex = 2;
            this->button1->Text = L"button1";
            this->button1->UseVisualStyleBackColor = true;
            this->button1->Click += gcnew System::EventHandler(this,
&MyForm::button1_Click);
            //
            // MyForm
            //
```

```
                this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                this->ClientSize = System::Drawing::Size(284, 261);
                this->Controls->Add(this->panel2);
                this->Controls->Add(this->panel1);
                this->Name = L"MyForm";
                this->Text = L"MyForm";
                this->Load += gcnew System::EventHandler(this, &MyForm::MyForm_Load);
                this->Resize += gcnew System::EventHandler(this,
&MyForm::MyForm_Resize);
                this->panel1->ResumeLayout(false);
                (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>pictureBox1))->EndInit();
                this->panel2->ResumeLayout(false);
                this->ResumeLayout(false);

            }

#pragma endregion
        private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e) {
            OpenFileDialog ^ openFileDialog1 = gcnew OpenFileDialog();
            if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)
            {
                pictureBox1->Image = Image::FromFile(openFileDialog1->FileName);
                // Примусово викликаємо обробник події Resize форми для встановлення
розмірів об'єктів panel1 та button1
                MyForm_Resize(nullptr, nullptr);
            }

        }

        private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
            this->Text = "Графічний файл";
            button1->Text = "Відкрити";
            button2->Text = "Зберегти";
            panel2->Height = 40;
            panel2->Dock = System::Windows::Forms::DockStyle::Bottom;
            button1->Dock = System::Windows::Forms::DockStyle::Right;
            panel1->Dock = System::Windows::Forms::DockStyle::Top;
            pictureBox1->SizeMode = PictureBoxSizeMode::AutoSize;
            panel1->AutoScroll = true; // Дозволяємо прокрутку зображення
            button2->Dock = System::Windows::Forms::DockStyle::Fill;
        }

        private: System::Void MyForm_Resize(System::Object^  sender, System::EventArgs^  e) {
            // Встановлення розмірів об'єктів panel1 та button1
            panel1->Height = this->Height - panel2->Height - 40;
            button1->Width = this->panel2->Width / 2;
        }

        private: System::Void button2_Click(System::Object^  sender, System::EventArgs^  e) {
            // Відобразити вікно SaveFileDialog щоб користувач міг зберегти зображення
            SaveFileDialog ^ saveFileDialog1 = gcnew SaveFileDialog();
            saveFileDialog1->Filter = "Зображення JPeg|*.jpg|Зображення
Bitmap|*.bmp|Зображення Gif|*.gif";
            saveFileDialog1->Title = "Зберегти файл зображення";
            saveFileDialog1->ShowDialog();
            // Якщо ім'я файлу не пустий рядок, зберегти зображення
            if (saveFileDialog1->FileName != "")
            {
                // Зберегти зображення через FileStream, створений методом OpenFile
```

```
                    System::IO::FileStream ^ fs =
safe_cast<System::IO::FileStream^>(saveFileDialog1->OpenFile());
                    // Можна зберегти зображення у відповідному форматі ImageFormat на
основі
                    // типу файлу, обраного в діалоговому вікні.
                    // Тип файлу береться з властивості FilterIndex
                    switch (saveFileDialog1->FilterIndex)
                    {
                    case 1:
                          this->pictureBox1->Image->Save(fs,
                                System::Drawing::Imaging::ImageFormat::Jpeg);
                          break;
                    case 2:
                          this->pictureBox1->Image->Save(fs,
                                System::Drawing::Imaging::ImageFormat::Bmp);
                          break;
                    case 3:
                          this->pictureBox1->Image->Save(fs,
                                System::Drawing::Imaging::ImageFormat::Gif);
                          break;
                    }
                    fs->Close();
             }
}
};
}
```

Лістинг 5.3

```
// Програма рисує обрану зі списку фігуру у формі

#pragma once

namespace WindowsForms {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Сводка для MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
		}

	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::ListBox^  listBox1;
	protected:

	private:
		/// <summary>
		/// Обязательная переменная конструктора.
		/// </summary>
		System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
		/// <summary>
		/// Требуемый метод для поддержки конструктора — не изменяйте
		/// содержимое этого метода с помощью редактора кода.
		/// </summary>
		void InitializeComponent(void)
		{
			this->listBox1 = (gcnew System::Windows::Forms::ListBox());
			this->SuspendLayout();
			//
			// listBox1
```

```
                    //
                    this->listBox1->FormattingEnabled = true;
                    this->listBox1->Location = System::Drawing::Point(126, 32);
                    this->listBox1->Name = L"listBox1";
                    this->listBox1->Size = System::Drawing::Size(120, 95);
                    this->listBox1->TabIndex = 0;
                    this->listBox1->SelectedIndexChanged += gcnew
                    System::EventHandler(this, &MyForm::listBox1_SelectedIndexChanged);
                    //
                    // MyForm
                    //
                    this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                    this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                    this->ClientSize = System::Drawing::Size(284, 261);
                    this->Controls->Add(this->listBox1);
                    this->Name = L"MyForm";
                    this->Text = L"MyForm";
                    this->Load += gcnew System::EventHandler(this, &MyForm::MyForm_Load);
                    this->Resize += gcnew System::EventHandler(this,
                                        &MyForm::MyForm_Resize);
                    this->ResumeLayout(false);

                }
#pragma endregion
        private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
                this->Text = "Обери графічний примітив";
                listBox1->Top = 0;
                listBox1->Left = 0;
                // Додаємо у об'єкт listBox1 перелік рядків
                // Це можна також зробити за допомогою властивості Items у вікні Properties
                listBox1->Items->AddRange(gcnew array <Object ^> {"Коло",
                        "Відрізок", "Прямокутник", "Сектор",
                        "Текст", "Еліпс", "Зафарбований сектор"});
                Font = gcnew System::Drawing::Font("Times New Roman", 14.F);
        }
        private: System::Void listBox1_SelectedIndexChanged(System::Object^  sender,
System::EventArgs^  e) {
                // Тут замість цієї події можна було б обробити
                // подію listBox1_Click.
                // Створення графічного об'єкта
                Graphics ^ Графіка = this->CreateGraphics();
                // Створення пера для малювання їм фігур
                Pen ^ Перо = gcnew Pen(Color::Red);
                // Створення пензля для "зафарбовування" фігур
                Brush ^ Пензель = gcnew SolidBrush(Color::Red);
                // Очищення області малювання шляхом її фарбування
                // в первісний колір форми
                Графіка->Clear(SystemColors::Control);
                // Або Графіка->Clear(Color::FromName("Control"));
                // Або Графіка->Clear(ColorTranslator::FromHtml("# EFEBDE"));
                // Задаємо координати лівого верхнього кута фігур
                int Ліво(listBox1->Width + 20), Верх(listBox1->Left + 20);
                // Задаємо ширину та висоту фігур, які відраховуються праворуч та вниз
                // від верхньої лівої точки
                int Ширина(this->Width - listBox1->Width - 60), Висота(this->Height - 80);
                // Задаємо діаметр для кола, сектора та зафарбованого сектора
                int Діаметр(Ширина);
                if (Ширина > Висота) Діаметр = Висота;
                // Задаємо параметри сектора
                int ПочатковийКут, КінцевийКут;
                Random^ rnd = gcnew Random();
```

```
                // Задаємо випадково початковий та кінцевий кут сектора
                // в інтервалі від 0 до 360 градусів
                ПочатковийКут = rnd->Next(0, 360);
                КінцевийКут = rnd->Next(0, 360);
                switch (listBox1->SelectedIndex) // Вибір фігури:
                {
                case 0: // - обрана окружність:
                        Графіка->DrawEllipse(Перо, Ліво, Верх, Діаметр, Діаметр);
                        break;
                case 1: // - обраний відрізок:
                        Графіка->DrawLine(Перо, Ліво, Верх, this->Width-40, this->Height-60);
                        break;
                case 2: // - обраний прямокутник:
                        Графіка->DrawRectangle(Перо, Ліво, Верх, Ширина, Висота);
                        break;
                case 3: // - обраний сектор:
                        Графіка->DrawPie(Перо, Ліво, Верх, Діаметр, Діаметр, ПочатковийКут,
                                КінцевийКут);
                        break;
                case 4: // - обраний текст:
                        Графіка->DrawString("Хто вчиться змолоду, \n" +
                                "не зазнає на старість голоду.",
                                Font, Пензель, Ліво, Верх);
                        break;
                case 5: // - обраний еліпс:
                        Графіка->DrawEllipse(Перо, Ліво, Верх, Ширина, Висота);
                        break;
                case 6: // - обраний зафарбований сектор:
                        Графіка->FillPie(Пензель, Ліво, Верх, Діаметр, Діаметр, ПочатковийКут,
                                КінцевийКут);
                        break;
                }
        }
        private: System::Void MyForm_Resize(System::Object^  sender, System::EventArgs^  e) {
                // Перемальовуємо фігуру при зміні розміру вікна
                listBox1_SelectedIndexChanged(nullptr, nullptr);
        }
};
}
```

Лістинг 5.4

```
// Програма зміни кольору форми за допомогою переліку наявних кольорів

#pragma once

namespace WindowsForms {
	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Сводка для MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
		}

	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::ComboBox^  comboBox1;
	protected:

	private:
		/// <summary>
		/// Обязательная переменная конструктора.
		/// </summary>
		System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
		/// <summary>
		/// Требуемый метод для поддержки конструктора — не изменяйте
		/// содержимое этого метода с помощью редактора кода.
		/// </summary>
		void InitializeComponent(void)
		{
			this->comboBox1 = (gcnew System::Windows::Forms::ComboBox());
			this->SuspendLayout();
			//
			// comboBox1
			//
```

```
                this->comboBox1->FormattingEnabled = true;
                this->comboBox1->Location = System::Drawing::Point(151, 12);
                this->comboBox1->Name = L"comboBox1";
                this->comboBox1->Size = System::Drawing::Size(121, 21);
                this->comboBox1->TabIndex = 0;
                this->comboBox1->SelectedIndexChanged += gcnew
                System::EventHandler(this, &MyForm::comboBox1_SelectedIndexChanged);
                //
                // MyForm
                //
                this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                this->ClientSize = System::Drawing::Size(284, 261);
                this->Controls->Add(this->comboBox1);
                this->Name = L"MyForm";
                this->Text = L"MyForm";
                this->Load += gcnew System::EventHandler(this, &MyForm::MyForm_Load);
                this->ResumeLayout(false);

        }
#pragma endregion
    private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
            // Отримуємо масив рядків імен кольорів з перерахування KnownColor.
            // Enum::GetNames повертає масив імен констант
            // в зазначеному перерахуванні:
            array <String ^> ^ ВсіКольори = Enum::GetNames(KnownColor :: typeid);
            // Видалення всіх елементів з колекції:
            comboBox1->Items->Clear();
            // Додаємо імена всіх кольорів в перелік comboBox1:
            for each (String ^ s in ВсіКольори)
                if (s != "Transparent") comboBox1->Items->Add(s);
            // Колір Transparent є "прозорим",
            // він не підтримується для форми
    }

    private: System::Void comboBox1_SelectedIndexChanged(System::Object^  sender,
System::EventArgs^  e) {
            // Обробка події зміни обраного
            // індексу в списку comboBox1:
            this->BackColor = Color::FromName(comboBox1->Text);
            // Напис в рядку заголовка форми:
            this->Text = "Колір:" + comboBox1->Text;
    }
    };
}
```

Лістинг 5.5

```
//Приклад побудови графіку за допомогою класу Graphics

#pragma once

namespace WindowsForms {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Сводка для MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
		}

	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::Panel^  panel1;
	private: System::Windows::Forms::Button^  button1;
	private: System::Windows::Forms::PictureBox^  pictureBox1;
	protected:

	private:
		/// <summary>
		/// Обязательная переменная конструктора.
		/// </summary>
		System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
		/// <summary>
		/// Требуемый метод для поддержки конструктора — не изменяйте
		/// содержимое этого метода с помощью редактора кода.
		/// </summary>
		void InitializeComponent(void)
		{
			this->panel1 = (gcnew System::Windows::Forms::Panel());
			this->button1 = (gcnew System::Windows::Forms::Button());
```

```
                    this->pictureBox1 = (gcnew System::Windows::Forms::PictureBox());
                    this->panel1->SuspendLayout();
                    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>pictureBox1))->BeginInit();
                    this->SuspendLayout();
                    //
                    // panel1
                    //
                    this->panel1->Controls->Add(this->button1);
                    this->panel1->Dock = System::Windows::Forms::DockStyle::Bottom;
                    this->panel1->Location = System::Drawing::Point(0, 207);
                    this->panel1->Name = L"panel1";
                    this->panel1->Size = System::Drawing::Size(284, 54);
                    this->panel1->TabIndex = 0;
                    //
                    // button1
                    //
                    this->button1->Dock = System::Windows::Forms::DockStyle::Fill;
                    this->button1->Location = System::Drawing::Point(0, 0);
                    this->button1->Name = L"button1";
                    this->button1->Size = System::Drawing::Size(284, 54);
                    this->button1->TabIndex = 0;
                    this->button1->Text = L"button1";
                    this->button1->UseVisualStyleBackColor = true;
                    this->button1->Click += gcnew System::EventHandler(this,
&MyForm::button1_Click);
                    //
                    // pictureBox1
                    //
                    this->pictureBox1->Dock = System::Windows::Forms::DockStyle::Fill;
                    this->pictureBox1->Location = System::Drawing::Point(0, 0);
                    this->pictureBox1->Name = L"pictureBox1";
                    this->pictureBox1->Size = System::Drawing::Size(284, 207);
                    this->pictureBox1->TabIndex = 1;
                    this->pictureBox1->TabStop = false;
                    //
                    // MyForm
                    //
                    this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                    this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                    this->ClientSize = System::Drawing::Size(284, 261);
                    this->Controls->Add(this->pictureBox1);
                    this->Controls->Add(this->panel1);
                    this->Name = L"MyForm";
                    this->Text = L"MyForm";
                    this->Load += gcnew System::EventHandler(this, &MyForm::MyForm_Load);
                    this->panel1->ResumeLayout(false);
                    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>pictureBox1))->EndInit();
                    this->ResumeLayout(false);
                }
#pragma endregion
                // Програма малює графік продажів по місяцях. Зрозуміло, що таким же чином
                // Можна побудувати будь-який графік по точках для інших прикладних цілей
                // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
                // Вихідні дані для побудови графіка (тобто вихідні точки):
                array <String ^> ^ Months;
                array <int> ^ Sales;
                Graphics ^ Графіка;
                // Далі, створюємо об'єкт Bitmap, який має
                // Той же розмір і роздільну здатність, що і PictureBox
```

```
            Bitmap ^ Растр;
            int ВідступЛіворуч, ВідступПраворуч, ВідступЗнизу, ВідступЗверху;
            int ДовжинаВертОсі, ДовжинаГоризОсі, УГоризОсі, ХВертОсі, Хmax, ХПочЕпюри;
            // Крок градуювання по горизонтальній і вертикальній осях:
            double ГоризКрок;
            int ВертКрок;
            // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

      private: System::Void button1_Click(System::Object ^ sender,
            System::EventArgs ^ e)
      {// Побудова графіку:
            Months = gcnew array <String ^> { "Січ", "Лют", "Бер", "Квіт", "Трав",
                  "Черв", "Лип", "Серп", "Вер", "Жовт", "Лист", "Груд"};
            Sales = gcnew array <int> {335, 414, 572, 629, 750, 931, 753, 599, 422, 301,
245, 155};
            ВідступЛіворуч = 35; ВідступПраворуч = 15;
            ВідступЗнизу = 20; ВідступЗверху = 10;
            Растр = gcnew Bitmap(pictureBox1->Width, pictureBox1->Height, pictureBox1-
>CreateGraphics());
            pictureBox1->BorderStyle = BorderStyle::FixedSingle;//Рамка навколо графіка
            УГоризОсі = pictureBox1->Height - ВідступЗнизу;
            ХВертОсі = pictureBox1->Width - ВідступЛіворуч;
            Хmax = pictureBox1->Width - ВідступПраворуч;
            ДовжинаГоризОсі = pictureBox1->Width - (ВідступЛіворуч + ВідступПраворуч);
            ДовжинаВертОсі = УГоризОсі - ВідступЗверху;
            ГоризКрок = (double)(ДовжинаГоризОсі / Sales->Length + 3);
            ВертКрок = (int)(ДовжинаВертОсі / 10);
            ХПочЕпюри = ВідступЛіворуч;//Початкове значення графіку по вісі X
                                    // Послідовно викликаємо наступні процедури:
            Графіка = Graphics::FromImage(Растр);
            МалюємоВісі();
            МалюємоГоризЛінії();
            МалюємоВертЛінії();
            МалюємоЕпюру();
            pictureBox1->Image = Растр;
            // Звільняємо ресурси, використовувані об'єктом класу Graphics:
            delete Графіка; // - Еквівалент C#: Графіка.Dispose
      } // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

      private: void МалюємоВісі()
      {
            Pen ^ Перо = gcnew Pen(Color::Black, 2);
            // Малювання вертикальної осі координат:
            Графіка->DrawLine(Перо, ВідступЛіворуч, УГоризОсі, ВідступЛіворуч,
ВідступЗверху);
            // Малювання горизонтальної осі координат:
            Графіка->DrawLine(Перо, ВідступЛіворуч, УГоризОсі, Хmax, УГоризОсі);
            Drawing::Font ^ Шрифт = gcnew Drawing::Font("Arial", 8);
            // Підписуємо значення по вісі Y
            for (int i = 1; i <= 10; i++)
            {// Малюємо "вусики" на вертикальній координатної осі:
                  int Y = УГоризОсі - i * ВертКрок;
                  Графіка->DrawLine(Перо, ВідступЛіворуч - 5, Y, ВідступЛіворуч, Y);
                  // Підписуємо значення продажів через кожні 100 одиниць:
                  Графіка->DrawString((i * 100).ToString(), Шрифт, Brushes::Black, 2, Y -
5.F);
            }
            // Підписуємо значення по вісі X
            for (int i = 0; i <= Months->Length - 1; i++)
            {// Малюємо "вусики" на горизонтальній координатної осі:
                  int X = ХПочЕпюри + (int)(ГоризКрок * (i + 1));
```

```
                        Графіка->DrawLine(Перо, X, YГоризОсі + 5, X, YГоризОсі);
                        // Підписуємо місяці на горизонтальній осі:
                        Графіка->DrawString(Months[i], Шрифт, Brushes::Black, ВідступЛіворуч -
10.F + (int)(i * ГоризКрок), YГоризОсі + 4.F);
                }
        } // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

        private: void МалюємоГоризЛінії()
        {
                Pen ^ ТонкеПеро = gcnew Pen(Color::LightGray, 1);
                for (int i = 1; i <= 10; i++)
                {// Малюємо горизонтальні майже "прозорі" лінії:
                        int Y = YГоризОсі - ВертКрок * i;
                        Графіка->DrawLine(ТонкеПеро, ВідступЛіворуч, Y, Xmax, Y);
                }
                delete ТонкеПеро; // - Еквівалент C#: ТонкоеПеро.Dispose();
        } // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

        private: void МалюємоВертЛінії()
        {// Малюємо вертикальні майже "прозорі" лінії
                Pen ^ ТонкеПеро = gcnew Pen(Color::Bisque, 1);
                for (int i = 0; i <= Months->Length - 1; i++)
                {
                        int X = ХПочЕпюри + (int)(ГоризКрок * i);
                        Графіка->DrawLine(ТонкеПеро, X, ВідступЗверху, X, YГоризОсі - 4);
                }
                delete ТонкеПеро;
        } // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

        private: void МалюємоЕпюру()
        {
                double ВертМасштаб = (double)ДовжинаВертОсі / 1000;
                // Значення ординат на екрані:
                array <int> ^ Y = gcnew array <int>(Sales->Length);
                // Значення абсцис на екрані:
                array <int> ^ X = gcnew array <int>(Sales->Length);
                for (int i = 0; i <= Sales->Length - 1; i++)
                {// Обчислюємо графічні координати точок:
                        Y[i] = YГоризОсі - (int)(Sales[i] * ВертМасштаб);
                        // Віднімаємо значення продажів, оскільки вісь Y екрану спрямована вниз
                        X[i] = ХПочЕпюри + (int)(ГоризКрок * i);
                }
                // Малюємо перше коло:
                Pen ^ Перо = gcnew Pen(Color::Blue, 3);
                Графіка->DrawEllipse(Перо, X[0] - 2, Y[0] - 2, 4, 4);
                for (int i = 0; i <= Sales->Length - 2; i++)
                {// Цикл по лініях між точками:
                        Графіка->DrawLine(Перо, X[i], Y[i], X[i + 1], Y[i + 1]);
                        // Віднімаємо 2, оскільки діаметр (ширина) точки = 4:
                        Графіка->DrawEllipse(Перо, X[i + 1] - 2, Y[i + 1] - 2, 4, 4);
                }
        } // ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e)
        {// Встановлення заголовку вікна та тексту кнопки при запуску вікна програми
                this->Text = "Побудова графіка";
                button1->Text = "Намалювати графік";
        }
};
}
```

Лістинг 5.6

```
// Програма малювання мишею у вікні

#pragma once

namespace WindowsForms {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Сводка для MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
		}

	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::Button^  button1;
	protected:

	private:
		/// <summary>
		/// Обязательная переменная конструктора.
		/// </summary>
		System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
		/// <summary>
		/// Требуемый метод для поддержки конструктора — не изменяйте
		/// содержимое этого метода с помощью редактора кода.
		/// </summary>
		void InitializeComponent(void)
		{
			this->button1 = (gcnew System::Windows::Forms::Button());
			this->SuspendLayout();
			//
			// button1
```

```
                    //
                    this->button1->Location = System::Drawing::Point(108, 226);
                    this->button1->Name = L"button1";
                    this->button1->Size = System::Drawing::Size(75, 23);
                    this->button1->TabIndex = 0;
                    this->button1->Text = L"button1";
                    this->button1->UseVisualStyleBackColor = true;
                    this->button1->Click += gcnew System::EventHandler(this,
&MyForm::button1_Click);
                    //
                    // MyForm
                    //
                    this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                    this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                    this->ClientSize = System::Drawing::Size(284, 261);
                    this->Controls->Add(this->button1);
                    this->Name = L"MyForm";
                    this->Text = L"MyForm";
                    this->Load += gcnew System::EventHandler(this, &MyForm::MyForm_Load);
                    this->MouseDown += gcnew
System::Windows::Forms::MouseEventHandler(this, &MyForm::MyForm_MouseDown);
                    this->MouseMove += gcnew
System::Windows::Forms::MouseEventHandler(this, &MyForm::MyForm_MouseMove);
                    this->MouseUp += gcnew System::Windows::Forms::MouseEventHandler(this,
&MyForm::MyForm_MouseUp);
                    this->ResumeLayout(false);

                }
#pragma endregion
                // Програма дозволяє при натиснутій лівій або правій кнопці миші
                // Малювати в формі
                // ~ ~ ~ ~ ~ ~ ~
                // Булева змінна ЧиМалювати дає дозвіл на малювання:
                bool ЧиМалювати;

        private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
                this->Text = "Малюю мишею у формі";
                button1->Text = "Стерти";
                ЧиМалювати = false; // на початку - не малювати
        }

        private: System::Void MyForm_MouseDown(System::Object^  sender,
System::Windows::Forms::MouseEventArgs^  e) {
                // Якщо натиснута кнопка миші - MouseDown, то малювати
                ЧиМалювати = true;
        }

        private: System::Void MyForm_MouseUp(System::Object^  sender,
System::Windows::Forms::MouseEventArgs^  e) {
                // Якщо кнопку миші відпустили, то НЕ малювати
                ЧиМалювати = false;
        }

        private: System::Void MyForm_MouseMove(System::Object^  sender,
System::Windows::Forms::MouseEventArgs^  e) {
                // Малювання прямокутника, якщо натиснута кнопка миші
                if (ЧиМалювати == true)
                {  // Намалювати прямокутник в точці (e.X, e.Y)
                    Graphics ^ Графіка = CreateGraphics();
                    Графіка->FillRectangle(gcnew SolidBrush(Color::Red), e->X, e->Y, 3, 3);
                    // 10x10 пікселів - розмір суцільного прямокутника
```

```
                    // e-> X, e-> Y - координати покажчика миші
                    delete Графіка; // Еквівалент C#: Графіка->Dispose();
                }
        }

        private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e) {
                // Методи очищення форми:
                Graphics ^ Графіка = CreateGraphics();
                Графіка->Clear(this->BackColor);
        }
};
}
```

Лістинг 5.7

```
//Приклад використання об'єктів TabControl, TabPage та відкриття файлів
//за допомогою діалогового вікна OpenFileDialog в об'єктах PictureBox та
//RichTextBox

#pragma once

namespace Windows_Forms {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;
	using namespace System::IO; //Підключення простору імен для методу
Path::GetFileName()

	/// <summary>
	/// Сводка для MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
		}

	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::TabControl^  tabControl1;
	protected:
	private: System::Windows::Forms::TabPage^  tabPage1;
	private: System::Windows::Forms::TabPage^  tabPage2;
	private: System::Windows::Forms::PictureBox^  pictureBox1;

	private: System::Windows::Forms::Button^  buttonВідкрити;
	private: System::Windows::Forms::RichTextBox^  richTextBox1;


	private:
		/// <summary>
		/// Обязательная переменная конструктора.
		/// </summary>
		System::ComponentModel::Container ^components;
```

```
#pragma region Windows Form Designer generated code
            /// <summary>
            /// Требуемый метод для поддержки конструктора — не изменяйте
            /// содержимое этого метода с помощью редактора кода.
            /// </summary>
            void InitializeComponent(void)
            {
                  this->tabControl1 = (gcnew System::Windows::Forms::TabControl());
                  this->tabPage1 = (gcnew System::Windows::Forms::TabPage());
                  this->pictureBox1 = (gcnew System::Windows::Forms::PictureBox());
                  this->tabPage2 = (gcnew System::Windows::Forms::TabPage());
                  this->richTextBox1 = (gcnew System::Windows::Forms::RichTextBox());
                  this->buttonВідкрити = (gcnew System::Windows::Forms::Button());
                  this->tabControl1->SuspendLayout();
                  this->tabPage1->SuspendLayout();
                  (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>pictureBox1))->BeginInit();
                  this->tabPage2->SuspendLayout();
                  this->SuspendLayout();
                  //
                  // tabControl1
                  //
                  this->tabControl1->Controls->Add(this->tabPage1);
                  this->tabControl1->Controls->Add(this->tabPage2);
                  this->tabControl1->Dock = System::Windows::Forms::DockStyle::Fill;
                  this->tabControl1->Location = System::Drawing::Point(0, 0);
                  this->tabControl1->Name = L"tabControl1";
                  this->tabControl1->SelectedIndex = 0;
                  this->tabControl1->Size = System::Drawing::Size(284, 261);
                  this->tabControl1->TabIndex = 0;
                  //
                  // tabPage1
                  //
                  this->tabPage1->Controls->Add(this->pictureBox1);
                  this->tabPage1->Location = System::Drawing::Point(4, 22);
                  this->tabPage1->Name = L"tabPage1";
                  this->tabPage1->Padding = System::Windows::Forms::Padding(3);
                  this->tabPage1->Size = System::Drawing::Size(276, 235);
                  this->tabPage1->TabIndex = 0;
                  this->tabPage1->Text = L"Графічний файл";
                  this->tabPage1->UseVisualStyleBackColor = true;
                  //
                  // pictureBox1
                  //
                  this->pictureBox1->Dock = System::Windows::Forms::DockStyle::Fill;
                  this->pictureBox1->Location = System::Drawing::Point(3, 3);
                  this->pictureBox1->Name = L"pictureBox1";
                  this->pictureBox1->Size = System::Drawing::Size(270, 229);
                  this->pictureBox1->TabIndex = 0;
                  this->pictureBox1->TabStop = false;
                  //
                  // tabPage2
                  //
                  this->tabPage2->Controls->Add(this->richTextBox1);
                  this->tabPage2->Location = System::Drawing::Point(4, 22);
                  this->tabPage2->Name = L"tabPage2";
                  this->tabPage2->Padding = System::Windows::Forms::Padding(3);
                  this->tabPage2->Size = System::Drawing::Size(276, 235);
                  this->tabPage2->TabIndex = 1;
                  this->tabPage2->Text = L"Файл *.rtf";
                  this->tabPage2->UseVisualStyleBackColor = true;
```

```
                    //
                    // richTextBox1
                    //
                    this->richTextBox1->Dock = System::Windows::Forms::DockStyle::Fill;
                    this->richTextBox1->Location = System::Drawing::Point(3, 3);
                    this->richTextBox1->Name = L"richTextBox1";
                    this->richTextBox1->Size = System::Drawing::Size(270, 229);
                    this->richTextBox1->TabIndex = 0;
                    this->richTextBox1->Text = L"";
                    //
                    // buttonВідкрити
                    //
                    this->buttonВідкрити->Dock = System::Windows::Forms::DockStyle::Bottom;
                    this->buttonВідкрити->Location = System::Drawing::Point(0, 238);
                    this->buttonВідкрити->Name = L"buttonВідкрити";
                    this->buttonВідкрити->Size = System::Drawing::Size(284, 23);
                    this->buttonВідкрити->TabIndex = 1;
                    this->buttonВідкрити->Text = L"Відкрити файл";
                    this->buttonВідкрити->UseVisualStyleBackColor = true;
                    this->buttonВідкрити->Click += gcnew System::EventHandler(this,
&MyForm::buttonВідкрити_Click);
                    //
                    // MyForm
                    //
                    this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                    this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                    this->ClientSize = System::Drawing::Size(284, 261);
                    this->Controls->Add(this->buttonВідкрити);
                    this->Controls->Add(this->tabControl1);
                    this->Name = L"MyForm";
                    this->Text = L"Перегляд файлів";
                    this->tabControl1->ResumeLayout(false);
                    this->tabPage1->ResumeLayout(false);
                    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>pictureBox1))->EndInit();
                    this->tabPage2->ResumeLayout(false);
                    this->ResumeLayout(false);
            }

#pragma endregion
private: System::Void buttonВідкрити_Click(System::Object^  sender, System::EventArgs^  e) {
      if (tabControl1->SelectedIndex == 0) //Перевірка чи активна перша вкладка
      {     //Тоді відкриваємо графічний файл
            //Створюємо об'єкт класу OpenFileDialog
            OpenFileDialog ^ openFileDialog1 = gcnew OpenFileDialog();
            //Встановлюємо в діалоговому вікні фільтр для відкриття файлів
            openFileDialog1->Filter = L"Файли зображень | *.bmp;*.jpg;*.gif; *.png";
            if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK
             && openFileDialog1->FileName->Length > 0)
            //Якщо користувач обрав в діалоговому вікні файл і його ім'я не пусте
            {
                  //Відкриваємо графічний файл в об'єкті pictureBox1
                  pictureBox1->Image = Image::FromFile(openFileDialog1->FileName);
                  //Розміщуємо ім'я відкритого файлу в заголовку вкладки
                  tabPage1->Text = Path::GetFileName(openFileDialog1->FileName);
            }
      }
      else //Активна друга вкладка
      {     //Тоді відкриваємо файл *.rtf
            //Створюємо об'єкт класу OpenFileDialog
            OpenFileDialog ^ openFileDialog1 = gcnew OpenFileDialog();
```

```
            //Встановлюємо в діалоговому вікні фільтр для відкриття файлів
            openFileDialog1->Filter = L"Текстові файли | *.rtf";
            if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK
             && openFileDialog1->FileName->Length > 0)
            //Якщо користувач обрав в діалоговому вікні файл і його ім'я не пусте
            {
                  //Відкриваємо файл формату RTF в об'єкті richTextBox1
                  richTextBox1->LoadFile(openFileDialog1->FileName);
                  //Розміщуємо ім'я відкритого файлу в заголовку вкладки
                  tabPage2->Text = Path::GetFileName(openFileDialog1->FileName);
            }
      }
}
};
}
```

Завдання до комп'ютерного практикуму №5

Варіанти

1. Обчислити та запам'ятати значення $z_i = F(x_i, y_i)$ в *n* різних точках (x_i, y_i), а потім знайти суму всіх отриманих значень функції. Прийняти $n > 15$; $x_{i+1} = x_i + \delta x$ ($x_0 = 2{,}2$; $\delta x = 1{,}8$); $y_{i+1} = y_i + \delta y$ ($y_0 = 1{,}6$; $\delta y = 0{,}2$).

$$F(x_i, y_i) = [\ln(x_i^2) - \ln(y_i^2)]\frac{\sqrt[3]{x_i - y_i}}{\operatorname{ctg}(x_i)}.$$

Побудувати графічну залежність $F = f(y_i)$.

2. Знайти перші десять та обчислити суму перших трьох, п'яти та дев'яти членів ряду, у якому

$$A_n = \frac{3n! - (-1)^n \cdot 2^{n-1}}{5^{n-1} + (n-1)!}$$

Побудувати графічну залежність $A = f(n)$.

3. Знайти коефіцієнт тепловіддачі *L* при конденсації пари на поверхні труби довжиною *H*=2м, яка утворює з горизонтальною поверхнею кут *b*= $\frac{\pi}{3}$. Температура стінки $t_{ст} = 210\ ^0C$. Температура пари змінюється в інтервалі $t = 140 - 250\ ^0C$ з кроком 10 ^0C.

$$L_b = L\sqrt{\sin b},$$

де *L* – коефіцієнт тепловіддачі для вертикальної труби, який розраховується за формулою:

$$L = 3 \cdot 10^{-3} \sqrt{\frac{H(t - t_{ст}) \cdot l^3 \cdot p^2 \cdot g}{r \cdot m^3}}$$

тут l = 0,652 Вт/(м*К) – теплопровідність конденсату;
p = 846 кг/м^3 - щільність конденсату;
g = 9,81 м/с^2 - прискорення вільного падіння;
r = 1,88*10^6 Дж/кг – теплота пароутворення;
m = 1,29*10^{-4} н*с/м^2 - динамічна в'язкість конденсату.

Побудувати графічну залежність $L = f(t)$.

4. Розрахувати константу швидкості реакції між йодистим метилом і диметил-п-толундіном у розчині нітробензолу.

$$k = \frac{\frac{x + x_p}{a}}{a \cdot t \cdot [1 - \left(\frac{x_p}{a}\right)^2]} \cdot \ln\left|\frac{\frac{x}{a}\left(1 - \frac{x + x_p}{a^2}\right) - 1}{a^2 \cdot \left(\frac{x_p}{a} - \frac{x}{a}\right)}\right|$$

де a = 0,006 – початкова концентрація;
x = 0,01246 - рівноважна концентрація;
x_p = 0,00675 – 0,01575, з кроком 0,001 – поточна концентрація;
$t = 12$.

Побудувати графічну залежність $k = f(x_p)$.

5. Розрахувати в'язкість діоксиду сірки для зміни температур $\mathrm{T} = 200 - 360\ ^0C$ з кроком 10 ^0C і атмосферному тиску, використавши формулу:

$$\mu = 6{,}3 \cdot 10^{-4} \cdot \frac{\mathrm{м}^{1/2} \cdot \mathrm{P}_{\mathrm{кр}}^{2/3}}{\mathrm{T}_{\mathrm{кр}}^{1/6}} \cdot \frac{\mathrm{T}_{\mathrm{прив}}^{3/2}}{\mathrm{T}_{\mathrm{прив}} \cdot 0{,}8}$$

де м = 64 – молекулярна вага SO_2;
$\mathrm{T}_{\mathrm{кр}}$ = 430 ^0C - критична температура;
$\mathrm{P}_{\mathrm{кр}}$ = 77,7 атм – критичний тиск;
$\mathrm{T}_{\mathrm{прив}}$ = (T + 273)/430 – приведена температура.
Надрукувати значення μ та $\sqrt{\mu}$.

Побудувати графічну залежність $\mu = f(\mathrm{T})$.

6. Розрахувати в'язкість аміаку в інтервалі температур від 10 до 22°C через 1°C за формулою Сатерленда

$$N = N_0 \frac{273{,}15 + C}{T \cdot C} \cdot (T/273{,}15)^n$$

При умові, що множник n визначається так:

$$n = A - B \cdot (t - 273{,}15) \cdot D \cdot 10^{-7} \cdot (t - 273{,}15)^2$$

де *A* = 1,06; *B* = 1,04; *D* = 0; *C* = 503; *N* = 91,6 * 10^{-7} Па*с.

Побудувати графічну залежність $N = f(T)$.

7. Розрахувати ізобарну теплоємність CO_2 в інтервалі температур від 20 до 150°C через 5°C за формулою:

$$\mathrm{C}_p = E + F \cdot (T/100) + G \cdot (T/100) + H \cdot (T/100) + N \cdot (T/100)$$

де *E* = 0,81513; *F* = 9,8454· 10^{-2}; *G* = -9,4747· 10^{-3}; *H* = 36,006· 10^{-5}; *N* = -0,0567.

Побудувати графічну залежність $\mathrm{C}_p = f(T)$.

8. Розрахувати ізобарну теплоємність аміаку в інтервалі температур від 40 до 170°C через 10°C за формулою:

$$\mathrm{C}_p = E + F \cdot (T/100) + G \cdot (T/100) + H \cdot (T/100) + N \cdot (T/100)$$

де *E* = 2,0183; *F* = 158,072 * 10^{-2}; *G* = 273,61 * 10^{-3}; *H* = -9380,9 * 10^{-5}; *N* = -1,9986,
Побудувати графічну залежність $\mathrm{C}_p = f(T)$.

9. Розрахувати ізохорну теплоємність CO_2 C_v в інтервалі температур від 30 до 160°C через 10°C за допомогою наступних залежностей

$$\mathrm{C}_p = E + F \cdot (T/100) + G \cdot (T/100) + H \cdot (T/100) + N \cdot (T/100),$$

де *E* = 0,81513; *F* = 9,8454· 10^{-2}; *G* = -9,4747· 10^{-3}; *H* = 36,006· 10^{-5}; *N* = -0,0567.

$$C_v = C_p - R/M,$$

де M – молекулярна маса газу; R – універсальна газова стала.
Побудувати графічну залежність $C_v = f(T)$.

10. Розрахувати ізохорну теплоємність аміаку C_v в інтервалі температур від 25 до 135°С через 5°С за допомогою наступних залежностей

$$C_p = E + F \cdot (T/100) + G \cdot (T/100) + H \cdot (T/100) + N \cdot (T/100),$$

де E = 2,0183; F = 158,072 * 10^{-2}; G = 273,61 * 10^{-3}; H = -9380,9 * 10^{-5}; N = -1,9986.

$$C_v = C_p - R/M,$$

де M – молекулярна маса аміаку; R – універсальна газова стала.

Побудувати графічну залежність $C_v = f(T)$.

11. Розрахувати в'язкість води в інтервалі температур від 16 до 27°С через 1°С за формулою:

$$N_o = A(B + t)^n,$$

де A = 0,59849, B = 43, 252, n = -1,5423.

Побудувати графічну залежність $N_o = f(t)$.

12. Розрахувати теплопровідність аміаку в інтервалі температур від 16 до 27°С через 1°С за формулою:

$$L = A\frac{(T_K + c)}{(T + C)} \cdot \frac{T^{\frac{3}{2}}}{T_K^{\frac{3}{2}} M^{\frac{5}{6}}},$$

де A = 4,3643 * 10^{-3}; C = 503; T_K = 239,73 К; M – молекулярна маса газу.

Побудувати графічну залежність $L = f(T)$.

13. Розрахувати густину повітря в інтервалі температур від 45 до 74°С за формулою:

$$R = \frac{1{,}293 \cdot P}{(1 + 0{,}00367 \cdot t) \cdot 760}, \text{кг/м}^3$$

де P – тиск, який дорівнює 105,2 кПа.

Побудувати графічну залежність $R = f(t)$.

14. Розрахувати швидкість осадження кулеподібної частинки радіусом r ($r = 0{,}1 \cdot 10^{-3}$ м, $r = 0{,}2 \cdot 10^{-3}$ м … $r = 1{,}3 \cdot 10^{-3}$ м) у газі. Використати формулу:

$$W_{oc} = \frac{d^2 \cdot R \cdot g}{18 \cdot M_c},$$

де g = 9,81 м^2/с, R = 1684 кг/м^3; M_c = 5,2 * 10^{-4} Н * с/м^2.

Побудувати графічну залежність $W_{oc} = f(r)$.

15. Розрахувати час процесу фільтрування суспензії об'ємом V ($V = 50 \cdot 10^{-6}$, $60 \cdot 10^{-6}$, $70 \cdot 10^{-6}$, $80 \cdot 10^{-6}$, … , $250 \cdot 10^{-6}$ м^3) за формулою:

$$t = \frac{M \cdot r_0 \cdot x_0}{2p} \cdot \frac{V^2}{S^2} + \frac{M \cdot R}{P} \cdot \frac{V}{S},$$

де M = $5{,}17 \cdot 10^{-9}$ Н * с/м^3; $r = 4{,}15 * 10^{12}$ м$^{-2}$; $R = 2{,}3 * 10^{10}$ м$^{-1}$; $x_0 = 0{,}178$ м^3/м^3; $P = 0{,}6 \cdot 10^5$ Н/м^2; $S = 1$ м.

Побудувати графічну залежність $t = f(V)$.

16. Розрахувати витрату потужності на перемішування для пропелерної мішалки діаметром d = [2,0, 2,2, 2,4, 2,6; 2,8; 3,0; 3,2, 3,4, 3,6 м]. Використати формулу

$$N = K_N \cdot n^3 \cdot R_C \cdot d_m^5,$$

де K_N = 0,32; n = 3,96 об/с; R_C = 1200 кг/м^3.

Побудувати графічну залежність $N = f(d)$.

17. Розрахувати коефіцієнт теплопровідності L нітробензолу в інтервалі температур від 40 до 110°С через 5°С. Використати формулу

$$L_t = L_{30} \cdot [1 - E \cdot (t - 30)],$$

де $L_{30} = \mathrm{A}_1 \cdot \mathrm{c} \cdot R \cdot \sqrt{\frac{R}{M}}$, $\mathrm{A}_1 = 4{,}22 \cdot 10^{-2}$; с=1,38$\cdot 10^3$ Дж/(кг*град); R = 1200 кг/м^3; M = 123; $\mathrm{E} = 1 \cdot 10^{-3}$ град$^{-1}$.

Побудувати графічну залежність $L_t = f(t)$.

18. Розрахувати значення густини та в'язкості сірчаної кислоти в інтервалі температур від 10 до 80°С через 5°С. Використати формулу

$$R = 1894{,}8 - 0{,}909 \cdot t; \qquad M = 1{,}406 \cdot 10^{-3} + 5{,}0087 \cdot 10^{-1}/t$$

Побудувати графічну залежність $R = f(t)$.

19. Розрахувати опір шару осаду при фільтруванні суспензії з діаметром частинок $d_{\text{ср}} = (0{,}1;\ 0{,}15;\ 0{,}20;\ 0{,}25;\ 0{,}3;\ 3{,}35;\ 0{,}4;\ 0{,}45;\ 0{,}5) * 10^{-3}$ м за формулою:

$$r_{\text{o}} = A \cdot d_{cp}^B,$$

де A = 9,8373*10; B = -0,3224.

Побудувати графічну залежність $r_{\text{o}} = f(d_{\text{ср}})$.

20. Розрахувати залежність продуктивності вакуумфільтру W (кг/(час*м^3)) від величини вакууму P (Н/м^2), який визначається від 20 до 150 мм рт. ст. з кроком P = 10 мм рт. ст. для d = 0,25 мм, якщо:

$$W = 7056 \cdot P^{0,5} \cdot d^2$$

Побудувати графічну залежність $W = f(\mathrm{P})$.

21. Розрахувати тиск пару хлористого метилену *P* (та $\ln P$) в інтервалі температур 10…60°С через 5°С, якщо

$$\ln P = -\frac{A}{T} + B \cdot \ln T + C,$$

де *A* = 1995,6; *B* = -3,4426; *T* = *t* + 273; *C* = 8,621.

Побудувати графічну залежність $\mathrm{P} = f(T)$.

22. Розрахувати швидкість *R* хімічної реакції *A* -> *B* у діапазоні температур T = 300…450К із кроком 10 К, якщо

$$R = \mathrm{K}_0 \cdot \exp\left(-\frac{E}{R} \cdot T\right) \cdot C_0,$$

де K_0 = 2,05 * c^{-1}; *R* = 1,98725 кал/моль*град; *E* = 20100 кал/моль; C_0 = 4 моль/$\mathrm{м}^3$.

Побудувати графічну залежність $R = f(T)$.

23. Розрахувати, який об’єм займе 10 г азоту при тиску *P* = 4,5 атм і зміні температури від 200 до 400 К із кроком 10 К.
Використати формулу 1; *R* = 0,082057 л*атм/моль*град.

Побудувати графічну залежність $V = f(T)$.

24. Розрахувати значення константи швидкості хімічної реакції

$$K = A \cdot \exp\left(\frac{-5{,}29 \cdot 10^4}{R \cdot T}\right)$$

В інтервалі температур 600 … 800 °С через 20°С, якщо А = 7,5· 10^{-5}.

Побудувати графічну залежність $K = f(t)$.

25. Розрахувати значення енергії Гіббса

$$G = -R \cdot T \cdot \ln K$$

хімічної реакції в інтервалі температур від 400 до 550 К із кроком 10 К, якщо $\mathrm{R} = 1{,}3143$ Дж/моль $*$ К, К = 9,237.

Побудувати графічну залежність $G = f(T)$.

26. Обчислити та запам’ятати значення:

$$y_{ij} = A_i^{2,2} + 1{,}5 \cdot A_i \cdot B_i \cdot C_j - B_i^{1,5} \cdot \sqrt{A_i + 2B_i}; \qquad A_i = 1{,}8 \cdot \cos^3 x_i; \qquad B_i = e^{2/x_i} + 2{,}6.$$

а потім знайти суму всіх отриманих значень функції y_{ij}.
Прийняти x_i = 2,4; 2,3; 2,2; 2,1; 2,0; … ; 1,2; C_j = 2,31.

Побудувати графічну залежність $y_{ij} = f(x_i)$.

27. Обчислити та запам'ятати значення $z_i = F(x_i, y_i)$ в *n* різних точках (x_i, y_i), а потім знайти суму всіх отриманих значень функції. Прийняти $n > 10$; $x_{i+1} = x_i + \delta x$ ($x_0 = 2.8$; $\delta x = 2.1$); $y_{i+1} = y_i + \delta y$ ($y_0 = 1.7$; $\delta y = 0.3$).

$$F(x_i, y_i) = [\ln(x_i^2) - \ln(y_i^2)]\frac{\sqrt[3]{x_i - y_i}}{\operatorname{ctg}(x_i)}.$$

Побудувати графічну залежність $F = f(x_i)$.

28. Обчислити значення:

$$y_i = a_i^{2b_i} + b_i^{3a_i}; \qquad a_i = e^{2,5/x_i} + 3x_i \cdot C; \qquad b_i = \sin^2 x_i + x_i^{0,5}.$$

Прийняти x_i = 0,3; 0,7; 0.9; 1,1; 1,2; 1.4; 1,7; 1,8; 1,9; 2.2; 2.5; 2.7; C=2,15.

Побудувати графічну залежність $y_i = f(x_i)$.

29. Визначити коефіцієнт дифузії двооксиду сірки у повітрі при температурі $t = 16 - 28\ ^0C$ з кроком 1 ^0C та P = 1 атм. Використати формулу:

$$D = 4{,}3 \cdot 10^{-3} \cdot \frac{T^{3/2}}{P \cdot (V_A^{1/3} - V_B^{1/3})^2} \cdot \sqrt{\frac{1}{M_A} + \frac{1}{M_B}}$$

де $V_A - V_{SO_2}$ = 44,8 см^3/моль;
$V_B - V_{BO3}$ = 29,9 см^3/моль;
T = 273 +t;
M_A = 64;
M_B = 28,95.

Побудувати графічну залежність $D = f(t)$.

30. Знайти перші десять та обчислити суму перших трьох, п'яти та дев'яти членів ряду, у якому

$$A_n = \frac{2n! - (-1)^n \cdot 2^{n-1}}{2^{n+1} + n!}$$

Побудувати графічну залежність $A = f(n)$.

Контрольні питання

1) Для чого використовується метод `OnPaint`?
2) Опишіть метод `DrawImage` та його параметри.
3) Для чого використовується об'єкт класу `Graphics`?
4) Опишіть призначення та принцип використання методів `DrawString`, `DrawLine` та `DrawEllipse` класу `Graphics`.
5) Для чого використовується об'єкт класу `PictureBox`? Опишіть його властивості.
6) Опишіть призначення та принцип роботи об’єкту класу `OpenFileDialog`. Які властивості він має?
7) Опишіть призначення та принцип роботи об’єкту класу `SaveFileDialog`. Які властивості він має?
8) Опишіть призначення властивості `Dock` для деяких візуальних об’єктів. Перерахуйте можливі значення даної властивості та очікуваний результат.
9) Для чого використовується властивість `Filter` для об’єктів класів `OpenFileDialog` та `SaveFileDialog`? Яким чином присвоюються значення для цієї властивості?
10) Для чого використовуються об’єкти класу `TabControl`? Поясніть принцип їх роботи.
11) Для чого використовуються об’єкти класу `TabPage`? Поясніть принцип роботи з ними.
12) Для чого застосовується функція `GetFileName()` та які особливості її використання?
13) Для чого використовуються функції `FromFile()` та `LoadFile()`.

Комп'ютерний практикум 6

Робота з базами даних

Мета: вивчити прийоми роботи з базами даних в середовищі CLR. Навчитись створювати бази даних формату *.mdb. Засвоїти прийоми редагування таблиць бази даних з використанням об'єкту DataGridView. Навчитись створювати прості запити на вибірку даних з таблиці.

Завдання: створити новий пустий проект CLR, додати в нього форму Windows Forms. Додати необхідні об'єкти з Панелі елементів для створення головного та контекстного меню, а також панелі інструментів. Додати до форми потрібні об'єкти класів Panel, Button, DataGridView для створення графічного інтерфейсу. Програмним шляхом створити базу даних формату MS Access *.mdb та таблицю в ній. Програмно створити структуру таблиці не менше ніж з 6 полів, вказавши поля та їх типи в новій таблиці. Розробити СКБД, яка дозволятиме переглядати та редагувати і додавати дані до БД з використанням об'єкту DataGridView. Наповнити таблицю БД інформацією, що міститиме не менше 30 записів. Передбачити поле для створення простих запитів на вибірку даних, в тому числі відбір записів за умовою та видалення записів за умовою.

Загальні вимоги.

1) Створити проект Windows Forms.
2) Додати в нього необхідні елементи керування та програмний код для реалізації поставленої задачі згідно отриманого варіанту завдання
3) Програмний код має бути чітко структурований.
4) Імена об'єктів мають нести сенсові навантаження.
5) Програмний код має супроводжуватись коментарями в тексті програми.

Вимоги до виконання.

1) Створити новий пустий проект CLR в Visual C++ з використанням створеного шаблону проектів Windows Forms з ім'ям виду Прізвище_КПР6.
2) Додати у форму об'єкт головного меню класу MenuStrip.
3) Додати у форму об'єкт контекстного меню класу ContextMenuStrip.
4) Додати у форму об'єкт панелі інструментів класу ToolStrip.
5) Додати об'єкт DataGridView для відображення таблиці бази даних.
6) Додати інші необхідні об'єкти для створення програмного інтерфейсу системи керування базою даних та запрограмувати їх роботу.

7) Створити структуру бази даних формату MS Access *.mdb та таблицю в ній праграмними засобами.
8) Відображення таблиці БД організувати за допомогою об'єкту DataGridView.
9) Наповнити базу даних з використанням створеного програмного модуля.
10) Дотати до програми можливість створення простих запитів на вибірку даних з таблиці, в тому числі відбір записів за умовою та видалення записів за умовою

Теоретичні відомості

6.1 Технологія ADO.NET

ADO.NET (ActiveX Data Objects .NET) — це набір бібліотек, що поставляється з Microsoft .NET Framework і призначений для взаємодії з різними сховищами даних з .NET-застосунків. Бібліотеки ADO.NET включають класи для приєднання до джерела даних, виконання запитів і обробки їхніх результатів. Крім того, ADO.NET можна використовувати в якості надійного, ієрархічно організованого, відокремленого кешу даних для автономної роботи з даними.

ADO.NET була розроблена компанією Microsoft, для вирішення проблем, які виникали при роботі з ADO та попередніми технологіями, такими як: Data Access Objects (DAO), Remote Data Objects (RDO). Основною перевагою ADO.NET застосунків є гнучкість та розвинута підтримка XML.

ADO.NET володіє багатьма перевагами порівняно з іншими технологіями доступу до даних. Основні з них це:

Підтримка XML - ADO також підтримує XML, але не буде так само ефективно обробляти XML-дані, як це робить ADO.NET, оскільки ADO.NET створювався з врахуванням XML, а ADO - ні.

Простота модифікації - протягом терміну служби системи в неї можна вносити незначні зміни, однак спроби провести архітектурні зміни трапляються рідко, через виняткову складність завдання. На жаль, при природному розвитку подій такі зміни іноді виявляються необхідними.

Простота програмування - компоненти даних ADO.NET в Visual Studio інкапсулюють функціональні можливості доступу до даних різними способами, що допомагає розробляти програмні продукти значно швидше і з меншою кількістю помилок.

Продуктивність - для непідключених застосунків набори даних ADO.NET дають виграш в продуктивності в порівнянні з непідключеними наборами записів ADO. Передача непідключеного набору записів між рівнями за допомогою COM - упаковки може призвести до великої витрати обчислювальних

ресурсів, тому що значення в наборі записів перетворюються до типів даних, відомих COM. У ADO.NET таке перетворення типів даних не потрібно.

Постачальник даних .NET — це набір класів, призначених для взаємодії зі сховищем даних певного типу. .NET Framework включає два постачальника - SQL Client.NET Data Provider і OLE DB.NET Data Provider. Постачальник OLE DB.NET Data Provider дозволяє взаємодіяти з різними сховищами даних за допомогою постачальника OLE DB. Постачальник SQL Client.NET Data Provider розрахований виключно на взаємодію з БД SQL Server. Кожен постачальник даних .NET реалізує однакові базові класи — Connection, Command, DataAdapter, DataReader, Parameter, Transaction тощо, конкретне ім'я яких залежить від постачальника. Так, у постачальника SQL Client.NET Data Provider є об'єкт SqlConnection, а у постачальника OLE DB.NET Data Provider — об'єкт OleDbConnection (табл. 6.1).

Таблиця 6.1 - Постачальники даних ADO.NET

Постачальник даних	Простір імен	Компоновочний блок
OLE DB	System.Data.OleDb	System.Data.dll
Microsoft SQL Server	System.Data.SqlClient	System.Data.dll
Microsoft SQL Server Mobile	System.Data.SqlServerCe	System.Data.SqlServerCe.dll
ODBC	System.Data.Odbc	System.Data.dll
Oracle	System.Data.OracleClient	System.Data.OracleClient.dll

Для розробників, які пишуть керований код, ADO.NET надає функціональний набір, подібний до функціонального набору, який надають об'єкти даних ActiveX (ADO) розробникам моделей об'єктів власних компонентів (COM). Для доступу до даних в додатку .NET ми рекомендуємо використовувати ADO.NET, а не ADO. ADO.NET надає найпряміший спосіб доступу до даних в .NET Framework.

6.2 Створення бази даних MS Access

Створимо програму, яка під час своєї роботи створює базу даних Access, тобто файл типу *.mdb. Ця база даних буде порожньою, тобто вона не буде містити жодної таблиці. Наповнювати базу даних таблицями можна згодом як з програмного коду Visual C++, так і використовуючи MS Access. В цьому прикладі технологія ADO.NET не використовується.

Запустимо Visual Studio і в вікні Створення проекту (New Project) виберемо в середовищі CLR вузла Visual C++ додаток шаблону Windows

Forms. Дамо ім'я новому проекту БазаДаних. Щоб додати до нашого проекту DLL-бібліотеки ADO потрібно виконати наступні дії.

1) У вікні Оглядача рішень (Solution Explorer) клацаємо мишею по імені проекту БазаДаних для його виділення.
2) В пункті меню Проект (Project) виберемо команду Додати посилання... (Add Reference...) (рис. 6.1).

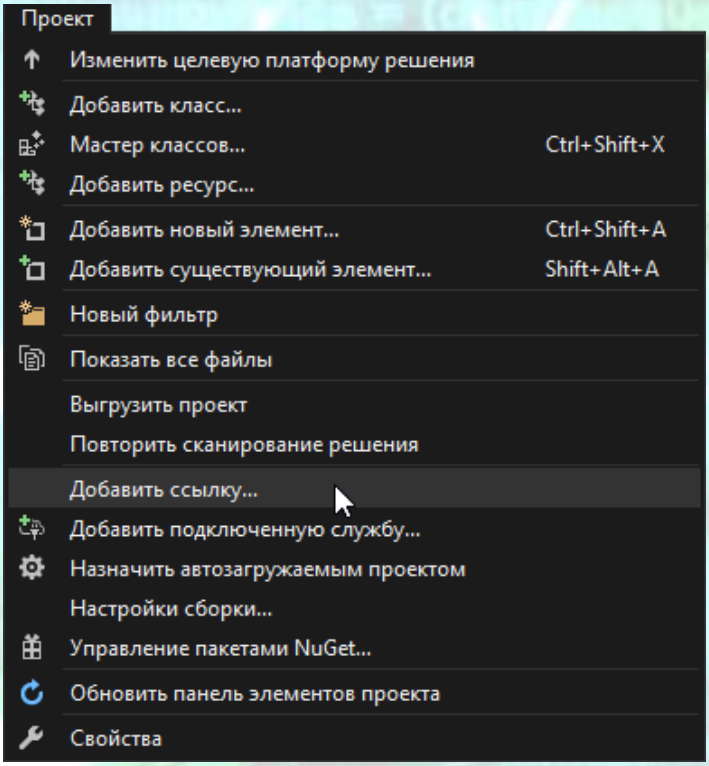

Рисунок 6.1 – Команда Додати посилання... меню Проект

3) На вкладці COM двічі клацнемо по посиланню Microsoft ADO Ext. 6.0 for DDL and Security (або більш ранню версію Microsoft ADO Ext. 2.8 for DDL and Security), додавши тим самим цю бібліотеку в поточний проект (рис. 6.2).
4) Переконатися в тому, що тепер існує посилання на цю бібліотеку, можна у вікні Властивості (Properties). Тут, клацнувши на вузлі Посилання (References), побачимо гілку ADOX (рис. 6.3). Тепер ми можемо посилатися на це ім'я в програмному коді.

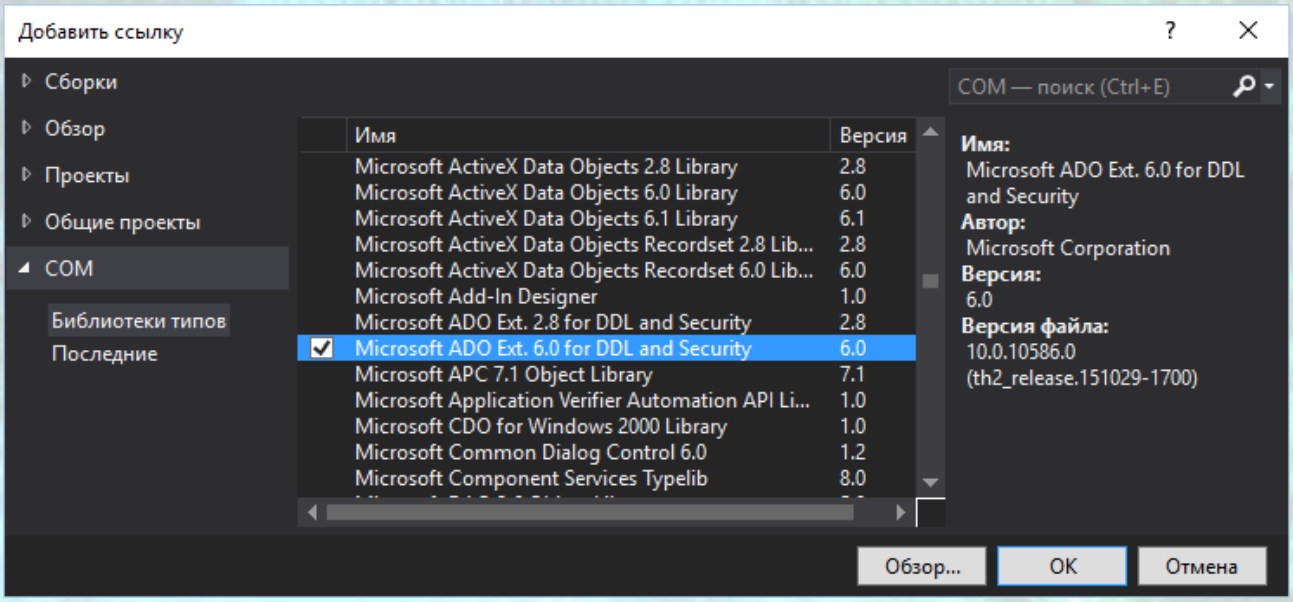

Рисунок 6.2 –Додавання бібліотеки Microsoft ADO Ext. 6.0 for DDL and Security

Далі переходимо у вікно конструктора форми і двічі клацнувши по формі, потрапляємо до обробника події MyForm_Load, де вводимо програмний код, наведений у лістингу нижче.

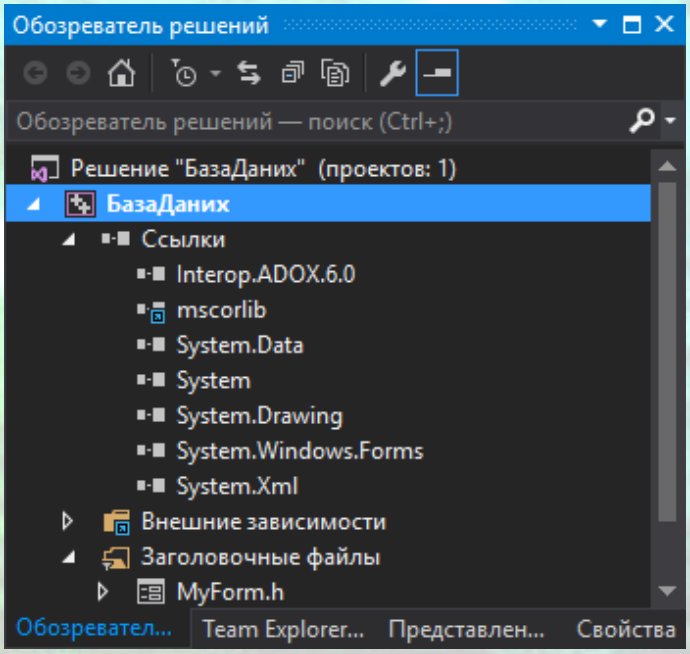

Рисунок 6.3 – Гілка ADOX у вузлі Посилання вікна Оглядача рішень

```
#pragma endregion
	private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
		ADOX::Catalog ^ Каталог = gcnew ADOX::Catalog();
		try
		{
			Каталог->Create("Provider=Microsoft.Jet." +
					"OLEDB.4.0;Data Source=d:\\Нова_БД.mdb");
			MessageBox::Show("База даних d:\\Нова_БД.mdb успішно створена",
```

```
            "Створення нової БД MS Access", MessageBoxButtons::OK, MessageBoxIcon::Information);
                }
                catch (System::Runtime::InteropServices::COMException ^ Ситуація)
                {
                    MessageBox::Show(Ситуація->Message, "Створення нової БД MS Access",
                                     MessageBoxButtons::OK, MessageBoxIcon::Warning);
                }
                finally
                { Каталог = nullptr; }
            }
            };
}
```

Програма працює наступним чином: створюємо екземпляр класу ADOX: `catalog`, одна з його функцій `Create` здатна створювати базу даних, якщо на її вхід подати рядок підключення. Зауважимо, що в рядок підключення входить також і повний шлях до створюваної БД. Функція `Create` укладена в блоки `try ... catch`, які обробляють виняткові ситуації. Після запуску цієї програми отримаємо повідомлення про створення бази даних (рис. 6.4).

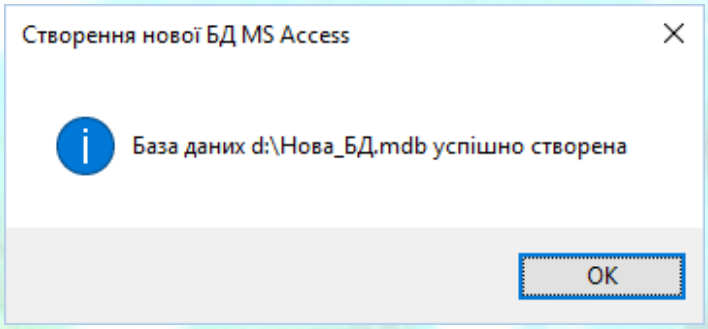

Рисунок 6.4 – Повідомлення про успішне створення нової БД

Якщо запустити наш додаток ще раз, то ми отримаємо повідомлення про те, що така база даних вже існує (рис. 6.5), оскільки БД `Нова_БД.mdb` щойно створена.

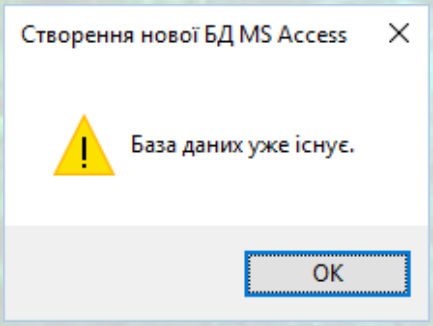

Рисунок 6.5 – Повідомлення про існування БД

Текст даного повідомлення генерувався оброблювачем виняткової ситуації `Ситуація->Message`. Ми додали власний заголовок вікна `"Створення`

нової БД MS Access", вказали, що в ньому має бути лише кнопка OK, та обрали тип вікна повідомлення Warning.

```
MessageBox::Show(Ситуація->Message, "Створення нової БД MS Access",
                                        MessageBoxButtons::OK, MessageBoxIcon::Warning);
```

6.3 Запис структури таблиці в порожню базу даних MS Access. Програмне підключення до БД

Тепер тут і далі ми будемо використовувати тільки найсучаснішу технологію ADO.NET. Створимо програму, яка записує структуру таблиці, тобто «шапку» таблиці, в існуючу БД. У цій БД може не бути жодної таблиці, тобто БД може бути порожньою. Або в БД можуть вже бути таблиці, але назва нової таблиці повинна бути унікальною.

Приклад програмної реалізації

Приклад програмного підключення до бази даних за допомогою об'єкту класу OleDbConnection з задаванням SQL-запиту в об'єкті класу OleDbCommand і виконанням запиту функцією ExecuteNonQuery().

Створимо базу даних Нова_БД.mdb в кореневому каталозі логічного диска D:, використовуючи MS Access або програмним шляхом, як це було показано в попередньому підрозділі. Ніякі таблиці в базі даних створювати не будемо, тобто наша БД буде порожньою. Тепер запустимо Visual Studio і в вікні Створення проекта (New Project) виберемо в середовищі CLR вузла Visual C++ додаток шаблону Windows Forms. Потім у обробник події MyForm_Load запишемо програмний код, представлений в лістингу нижче.

```
#pragma endregion
private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
       // ЗАПИС СТРУКТУРИ ТАБЛИЦІ у порожню БД:
       // Створення примірника об'єкта OleDbConnection із зазначенням рядка підключення:
       OleDbConnection ^ Підключення = gcnew OleDbConnection(
                    "Provider = Microsoft.Jet.OLEDB.4.0; Data Source = D:\\Нова_БД.mdb");
       // Відкриття підключення:
       Підключення->Open();
       // Створення примірника об'єкта класу Command
       // Із завданням SQL-запиту:
       OleDbCommand ^ Команда = gcnew OleDbCommand("CREATE TABLE [БД телефонів]" +
                           " ([Номер п/п] counter, [ПІБ] char(20), " +
                           " [Номер телефону] char(20))", Підключення);
       try // Виконання команди SQL:
       {
              Команда->ExecuteNonQuery();
              MessageBox::Show("Структура таблиці 'БД телефонів' записана в порожню БД",
                           "Створення структури таблиці MS Access", MessageBoxButtons::OK,
                           MessageBoxIcon::Information);
       }
```

```
        catch (Exception ^ Ситуація)
        {
                MessageBox::Show(Ситуація->Message, "Створення структури таблиці MS Access",
                        MessageBoxButtons::OK, MessageBoxIcon::Warning);
        }
        Підключення->Close();
}
};
}
```

В програмний код ми також додали наступну директиву для більш короткого звернення до класів обробки даних:

```
using namespace System::Data::OleDb;
```

Потрібно звернути увагу, що оператор

```
OleDbCommand ^ Команда = gcnew OleDbCommand("CREATE TABLE [БД телефонів]" +
                        " ([Номер п/п] counter, [ПІБ] char(20), " +
                        " [Номер телефону] char(20))", Підключення);
```

формує SQL-запит у середині подвійних лапок у вигляді рядка тексту і в наслідок цього, на відміну від звичайних операторів C++, даний рядок не можна довільно розбивати на частини. Розбити оператор на кілька рядків можна лише за допомогою оператора *конкатенації +*, узявши кожну частину рядка SQL-запиту у подвійні лапки, як і було зроблено у операторі вище. У вікні редактору коду цей рядок можна не розбивати і записати разом. Тоді SQL-запит у подвійних лапках матиме вигляд

```
"CREATE TABLE [БД телефонів] ([Номер п/п] counter, [ПІБ] char(20), [Номер телефону] char(20))"
```

що набагато зрозуміліше, ніж у лістингу програмного коду.

Як видно з тексту програми, спочатку ми створюємо екземпляр класу `OleDbConnection` із зазначенням рядка підключення, це дозволить нам керувати цим рядком програмно. Далі створюємо екземпляр класу `OleDbCommand` із завданням SQL-запиту. У цьому запиті створюємо (CREATE) нову таблицю з ім'ям `БД телефонів` з трьома полями: `Номер п/п` типу лічильник (`counter`), `ПІБ` і `Номер телефону`. Ім'я таблиці і імена полів укладені в квадратні дужки, оскільки вони містять пробіли.

Щоб виконати цю SQL-команду, викликаємо метод `ExecuteNonQuery()`, який укладемо в блоки `try ... catch` для обробки виняткових ситуацій. Якщо SQL- запит успішно виконався, то отримуємо повідомлення: «`Структура таблиці 'БД телефонів' записана в порожню БД`» (рис. 6.6).

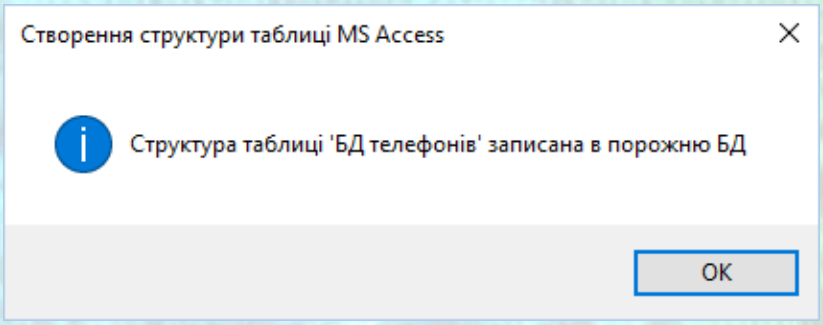

Рисунок 6.6 – Повідомлення про успішне створення структури таблиці БД

А якщо, наприклад, таблиця з таким ім'ям вже є в базі даних, то управління передається блоку `catch` (перехоплення виняткової ситуації), і ми отримуємо повідомлення про те, що така таблиця бази даних вже існує (рис. 6.7).

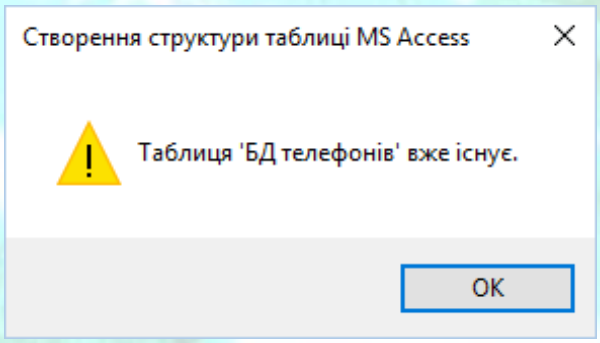

Рисунок 6.7 – Повідомлення про існування таблиці БД

Таким чином, в даній програмі спочатку організовано за допомогою об'єкту `Підключення` класу `OleDbConnection` підключення до БД через рядок підключення і відкриття підключення `Open()`. Потім здійснюється завдання SQL-запиту в об'єкті `Команда` класу `OleDbCommand` і виконання запиту функцією `ExecuteNonQuery()`. Якщо зв'язування даних організувати програмно, то ми отримаємо велику гнучкість для тих випадків, коли, наприклад, на стадії розробки невідомо заздалегідь, де (на якому диску, в якій папці) буде перебувати БД.

6.4 Додавання записів в таблицю бази даних MS Access

Зовсім маленьку програму з попереднього підрозділу можна використовувати для виконання будь-якого запиту, зверненого до бази даних. Наприклад, модифікуємо всього лише один рядок програмного коду програми з попереднього прикладу для додавання нового запису в таблицю БД. Для цього при створенні екземпляра об'єкта `OleDbCommand` задамо SQL-запит на вставку (`INSERT`) нового запису в таблицю БД.

Зауважимо, що в SQL-запиті ми свідомо звернулися до таблиці по імені `[бд телефонів]`, тобто з малої літери, хоча варто було б з великої. Справа в тому, що

в іменах таблиць слід точно вказувати регістр символу, оскільки їх пошук ведеться з урахуванням регістру (case-sensitive search). Однак це не обов'язково при наявності тільки однієї таблиці з таким ім'ям, оскільки при цьому використовується пошук без урахування регістру (case-insensitive search).

Властивості Connection об'єкта класу Command слід дати посилання на об'єкт класу OleDbConnection:

```
Команда->Connection = Підключення;
```

Причому для додавання запису в таблицю БД таке посилання обов'язкове на відміну від попереднього прикладу, де ми створювали нову таблицю в існуючій БД.

Програмний код буде виглядати так, як представлено в лістингу нижче.

```
#pragma endregion
	private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
		// Створення примірника об'єкта OleDbConnection
		// із зазначенням рядка підключення:
		auto Підключення = gcnew OleDbConnection(
			"Provider = Microsoft.Jet.OLEDB.4.0; Data Source = D:\\Нова_БД.mdb");
		// Відкриття підключення:
		Підключення->Open();
		// Створення примірника об'єкта OleDbCommand із завданням SQL-запиту:
		auto Команда = gcnew OleDbCommand(
			"INSERT INTO [бд телефонів] (" +
			"ПІБ, [номер телефону]) VALUES ('Оксана', '521-61-41')");
		// Для додавання запису в таблицю БД ця команда обов'язкова:
		Команда->Connection = Підключення;
		// Виконання команди SQL:
		Команда->ExecuteNonQuery();
		MessageBox::Show("В таблицю 'БД телефонів' додано запис", "Додавання записів
			в таблицю", MessageBoxButtons::OK, MessageBoxIcon::Information);
		Підключення->Close();
	}
	};
}
```

Найчастіше при відлагоджуванні програмного коду у Visual Studio C++, при роботі з БД з'являється необхідність перевірки роботи програми, наприклад потрібно дізнатися, чи створилася таблиця в БД, чи додався запис в таблицю БД, чи правильно сформований SQL-запит. Не обов'язково запускати MS Access, щоб виконати SQL-запит або перевірити правильність його синтаксису. Це можна зробити в середовищі Visual Studio. Для цього в пункті меню Вид (View) вибираємо команду Оглядач серверів (Other Windows Server Explorer) (комбінація клавіш Ctrl + Alt + S). Далі натискаємо кнопку Підключитися до бази даних. Після цього в діалоговому вікні Вибір джерела даних обираємо пункт Файл бази даних Microsoft Access та натискаємо кнопку Продовжити (рис. 6.8).

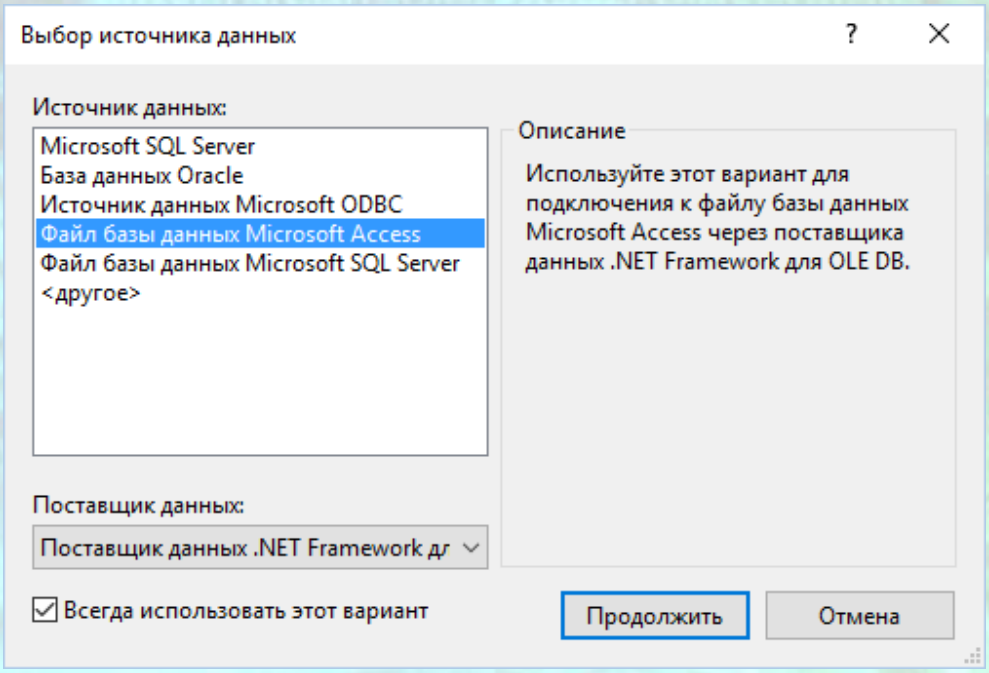

Рисунок 6.8 – Вікно вибору джерела даних

Далі у вікні Додати підключення у полі Ім’я файлу бази даних: вказуємо розташування нашої бази даних (рис. 6.9).

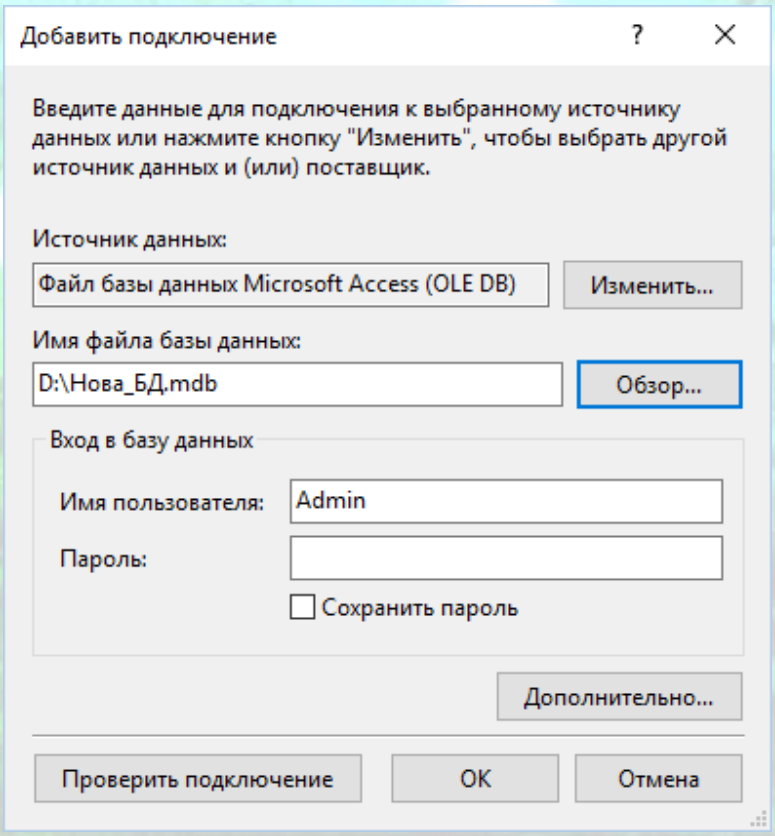

Рисунок 6.9 – Вікно додавання підключення

За допомогою кнопки Перевірити підключення можна протестувати вірність виконаних дій. Якщо база даних підключена, виведеться відповідне повідомлення (рис. 6.10).

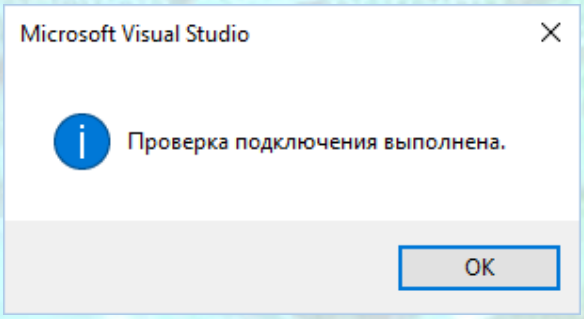

Рисунок 6.10 – Повідомлення про успішність підключення

Після успішного підключення бази даних у вікні Оглядача серверів з'явиться наша база даних у групі Підключення даних (рис. 6.11).

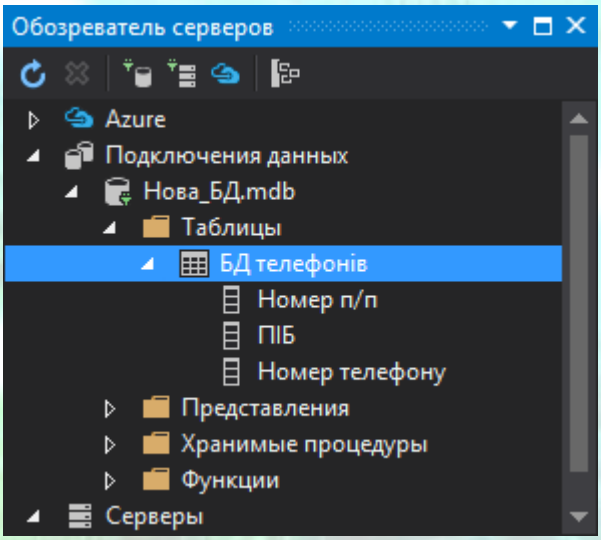

Рисунок 6.11 – Оглядач серверів з підключеною базою даних

Надалі, клацнувши правою кнопкою на імені нашої бази даних у вікні Оглядача серверів, ми можемо викликати контектсне меню, в якому є пункт для створення нового запиту (рис. 6.12).

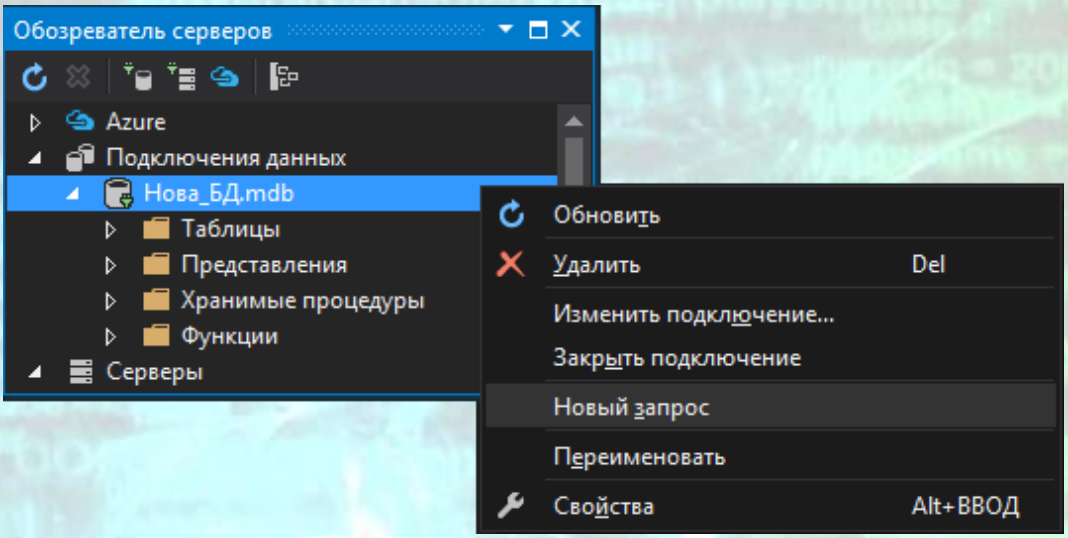

Рисунок 6.12 – Вибір пункту Новий запит у вікні Оглядача серверів

Після цього відкриється діалогове вікно Додати таблицю в якому можна обирати таблиці для формування запиту у вікні запитів. У вікні запитів ми можемо задавати SQL-запит, а потім, наприклад, клацаючи правою кнопкою миші, або перевіряти його синтаксис, або виконувати (Ctrl+R) (рис. 6.13).

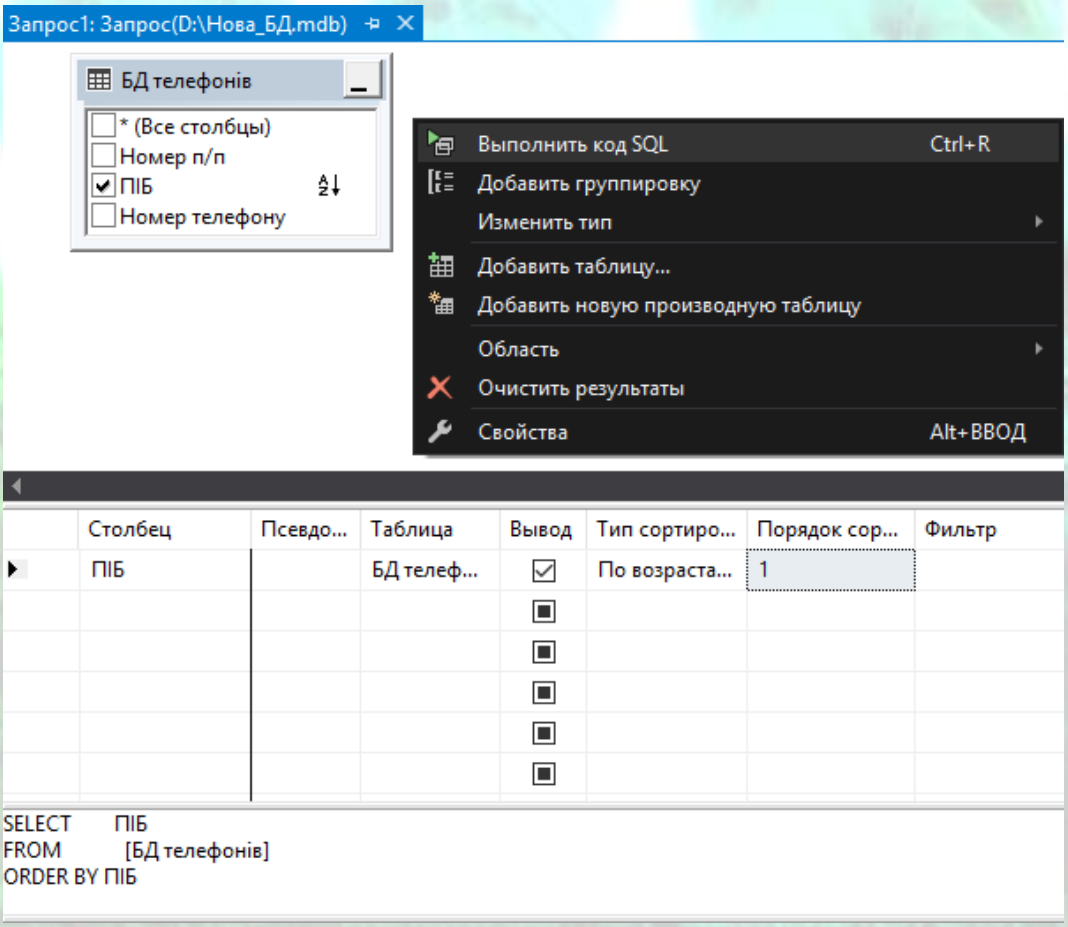

Рисунок 6.13 – Фрагмент вікна Запит з контекстним меню

6.5 Зчитування всіх записів з таблиці бази даних за допомогою об'єктів класів Command, DataReader та елементу керування DataGridView

В наступному прикладі розглянемо те, як можна вивести таблицю бази даних в елемент управління DataGridView (сітка даних, тобто таблиця даних) з використанням об'єктів класів Command та DataReader.

Приклад програмної реалізації

Приклад виведення існуючої таблиці БД в об'єкт класу DataGridView за допомогою об'єктів класів Command та DataReader.

Для вирішення цієї задачи створимо новий додаток за допомогою шаблону Windows Forms з середовища CLR вузла Visual C++. З Панелі елементів (Toolbox) додамо до форми елемент управління DataGridView та розтягнемо на всю форму за допомогою властивості Dock->Fill. Після цього запишемо наступний програмний код до обробника події MyForm_Load форми.

```
#pragma endregion
	private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
		// Створюємо об'єкт OleDbConnection і передаємо йому рядок підключення:
		auto Підключення = gcnew OleDb::OleDbConnection("Data Source = " +
		"D:\\Нова_БД.mdb; User ID = Admin; Provider = Microsoft.Jet.OLEDB.4.0;");
		Підключення->Open();
		// Створюємо об'єкт OleDbCommand, передаючи йому SQL-команду
		auto Команда = gcnew OleDb::OleDbCommand("Select * From [БД телефонів]",
			Підключення);
		// Виконуємо SQL-команду
		auto Читач = Команда->ExecuteReader();
		auto Таблиця = gcnew DataTable();
		// Заповнення "шапки" таблиці
		Таблиця->Columns->Add(Читач->GetName(0));
		Таблиця->Columns->Add(Читач->GetName(1));
		Таблиця->Columns->Add(Читач->GetName(2));
		while (Читач->Read() == true)
			// Заповнення клітин (комірок) таблиці
			Таблиця->Rows->Add(Читач->GetValue(0),
				Читач->GetValue(1), Читач->GetValue(2));
		// Тут три поля: 0, 1 і 2
		Читач->Close();
		Підключення->Close();
		dataGridView1->DataSource = Таблиця;
	}
	};
}
```

Виконання даної програми має на увазі, що існує база даних D:\Нова_БД.mdb, в якій є таблиця БД телефонів та в ній є не менше ніж три

поля. При виконанні цих умов, записи таблиці бази даних відобразяться в об’єкті dataGridView1 форми після запуску додатку на виконання.

Після виконання SQL-команди створюємо об'єкт Таблиця класу DataTable, який в кінці програмного коду задаємо як джерело (DataSource) для сітки даних dataGridView1. Заповнюємо «шапку» таблиці, тобто назви колонок, методом Add.

Далі в циклі While заповнюємо елементи таблиці. Фрагмент роботи програми показаний на рис. 6.14.

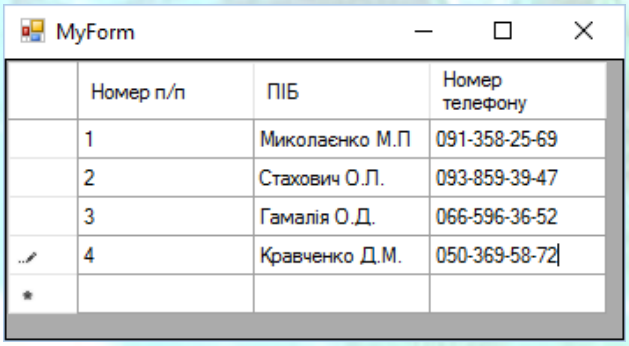

Рисунок 6.14 – Приклад роботи програми з відображення таблиці даних

У цій таблиці ми можемо впорядкувати записи по кожній із колонок, клацаючи мишею на назвах відповідних колонок. Чи можемо редагувати (змінювати) вміст комірок, але в базу даних ці зміни не потраплять (збереження не відбудеться).

Одна з ключових переваг використання об'єкта DataReader - це його швидкодія та використання невеликої кількості оперативної пам'яті. Однак застосування циклічного зчитування даних зводить ці переваги нанівець.

6.6 Зчитування даних з БД в сітку даних DataGridView з використанням об’єктів класів Command, Adapter та DataSet

Розглянемо приклад читання таблиці за допомогою об'єкта Adapter з бази даних шляхом вибору потрібних даних і передачі їх об'єкту DataSet. Дуже зручно прочитати таблицю, записану в DataSet, використовуючи елемент управління DataGridView (сітка даних, тобто таблиця даних), вказавши в якості джерела даних для сітки DataGridView об'єкт класу DataSet.

Приклад програмної реалізації

Приклад виведення існуючої таблиці БД в об’єкт класу DataGridView за допомогою об’єктів класів Command, Adapter та DataSet.

Створимо новий додаток з шаблону Windows Forms середовища CLR вузла Visual C++. З Панелі елементів (Toolbox) перетягнемо до форми елемент управління DataGridView та заповнимо ним всю форму за допомогою властивості Dock->Fill. Після цього запишемо наступний програмний код до обробника події MyForm_Load форми.

```
#pragma endregion
	private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
		this->Text = "Читання таблиці з БД:";
		auto Підключення = gcnew OleDb::OleDbConnection(
			"Data Source = D:\\Нова_БД.mdb; User ID =" +
			" Admin; Provider = Microsoft.Jet.OLEDB.4.0;");
		Підключення->Open();
		auto Команда = gcnew OleDb::OleDbCommand(
			"Select * From [БД телефонів]", Підключення);
		// Вибираємо з таблиці тільки ті записи, поле ПІБ яких
		// починається на букву "М":
		// Auto Команда =
		// gcnew OleDb::OleDbCommand("SELECT * FROM " +
		// "[БД телефонів] WHERE (ПІБ LIKE 'м%')", Підключення);
		// Створюємо об'єкт класу Adapter і виконуємо SQL-запит
		auto Адаптер = gcnew OleDb::OleDbDataAdapter(Команда);
		// Створюємо об'єкт класу DataSet
		auto НаборДаних = gcnew DataSet();
		// Заповнюємо DataSet результатом SQL-запиту
		Адаптер->Fill(НаборДаних, "БД телефонів");
		// Вміст DataSet у вигляді рядка XML для налагодження:
		auto РядокXML = НаборДаних->GetXml();
		// Вказуємо джерело даних для сітки даних:
		dataGridView1->DataSource = НаборДаних;
		// Вказуємо ім'я таблиці в наборі даних:
		dataGridView1->DataMember = "БД телефонів";
		Підключення->Close();
	}
	};
}
```

Як видно з тексту програми, спочатку ми створили об'єкт класу OleDbConnection, передаючи рядок підключення. Потім, створюючи об'єкт класу OleDbCommand, задаємо SQL-команду вибору всіх записів з таблиці БД телефонів. Тут ми можемо поставити будь-яку SQL-команду. У коментарі наведено приклад такої команди, яка містить SELECT і LIKE, в якій пропонується вибрати з таблиці БД телефонів тільки записи, в яких поле ПІБ починається на «м». Оператор LIKE використовується для *пошуку за шаблоном* (pattern matching) разом з *символами універсальної підстановки* (метасимвол) «зірочка» (*) і «знак питання» (?). Рядок шаблону укладений в апострофи.

Зауважимо також, що більшість баз даних використовує символ % замість значка * в LIKE-виразах.

Далі при створенні об'єкта класу `OleDbDataAdapter` виконуємо SQL-команду і при виконанні методу `Fill` заповнюємо об'єкт класу `DataSet` таблицею, отриманою в результаті SQL-запиту. Потім вказуємо як джерело даних для сітки даних `dataGridView1` об'єкт класу `DataSet`. Цього виявляється достатнім для виведення на екран результатів SQL-запиту рис. 6.15.

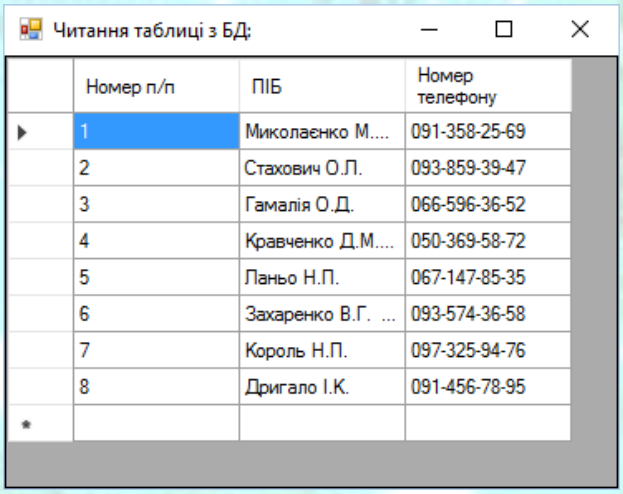

Рисунок 6.15 – Приклад роботи програми з відображення таблиці даних

Так само як і при використанні об'єкта класу `DataReader` в попередньому прикладі, в отриманій таблиці ми можемо сортувати записи за кожною з колонок. Можемо редагувати (змінювати) вміст комірок, але в базу даних ці зміни не потраплять (збереження не відбудеться).

Зауважимо, що тут за допомогою візуального проектування виконано тільки перетягування в форму сітки даних `DataGridView`, інше зроблено програмно, що забезпечує більшу гнучкість програми.

6.7 Оновлення записів в таблиці бази даних MS Access

Однією з основних чотирьох дій над даними в БД (`Select`, `Insert`, `Update` і `Delete`) є модифікація (`Update`, оновлення) даних. Створимо маленьку програму для оновлення записів в таблиці бази даних, але з великою зручністю (гнучкістю) управління програмним кодом.

Вже згадана в даному прикладі програма має форму, сітку даних `DataGridView`, в яку з бази даних зчитується таблиця при натисканні кнопки `Читати з БД`. Користувач має можливість *редагувати* дані в цій таблиці, після чого, при натисканні кнопки `Зберегти в БД`, дані в базі даних будуть *модифіковані*, тобто замінені новими.

Приклад програмної реалізації

Приклад модифікації записів БД.

Для цієї задачи створимо новий додаток з шаблону Windows Forms середовища CLR вузла Visual C++. З Панелі елементів (Toolbox) додамо до форми елемент Panel, в який помістимо два елементи кнопок Button. Також перетягнемо до форми ще один елемент Panel та помістимо в нього елемент управління DataGridView. Після цього запишемо наступний програмний код.

```
#pragma endregion

        // Програма оновлює записи (Update) в таблиці бази даних MS Access
        // ~ ~ ~ ~ ~ ~ ~ ~
        // Оголошуємо ці змінні поза всіма процедур, щоб
        // Вони були видні з будь-якої з процедур:
        DataSet ^ НабірДаних;
        OleDb::OleDbDataAdapter ^ Адаптер;
        OleDb::OleDbConnection ^ Підключення;
        OleDb::OleDbCommand ^ Команда;

private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
        this->panel1->Height = this->button1->Height*2;
        this->button2->Dock = System::Windows::Forms::DockStyle::Bottom;
        this->button1->Dock = System::Windows::Forms::DockStyle::Bottom;
        this->panel1->Dock = System::Windows::Forms::DockStyle::Bottom;
        this->panel2->Dock = System::Windows::Forms::DockStyle::Fill;
        this->dataGridView1->Dock = System::Windows::Forms::DockStyle::Fill;
        this->Text = "Редагування БД";
        НабірДаних = gcnew DataSet();
        Підключення = gcnew OleDb::
                OleDbConnection("Data Source=D:\\Нова_БД.mdb; User ID=" +
                        "Admin; Provider=Microsoft.Jet.OLEDB.4.0;");
        Команда = gcnew OleDb::OleDbCommand();
        button1->Text = "Читати з БД"; button1->TabIndex = 0;
        button2->Text = "Зберегти в БД";
        }

private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e) {
        // Читати з БД:
        if (Підключення->State ==
                ConnectionState::Closed) Підключення->Open();
        Адаптер = gcnew OleDb::OleDbDataAdapter(
                "Select * From [БД телефонів]", Підключення);
        // Заповнюємо DataSet результатом SQL-запиту
        Адаптер->Fill(НабірДаних, "БД телефонів");
        // Вміст DataSet у вигляді рядка XML для відладки:
        String ^ РядокXML = НабірДаних->GetXml();
        // Вказати джерело даних для сітки даних:
        dataGridView1->DataSource = НабірДаних;
        // Вказуємо ім'я таблиці в наборі даних:
        dataGridView1->DataMember = "БД телефонів";
        Підключення->Close();
        }
```

```
private: System::Void button2_Click(System::Object^  sender, System::EventArgs^  e) {
        // Зберегти в базі даних
        Команда->CommandText = "UPDATE [БД телефонів] SET [Номер телефону] =?," +
                " ПІБ =? WHERE ([Номер п/п] =?)";
        // Ім'я, тип і довжина параметра
        Команда->Parameters->Add("Номер телефону",
                OleDb::OleDbType::VarWChar, 50, "Номер телефону");
        Команда->Parameters->Add(
                "ПІБ", OleDb::OleDbType::VarWChar, 50, "ПІБ");
        Команда->Parameters->Add
                (gcnew OleDb::OleDbParameter("Original_Номер_п_п",
                        OleDb::OleDbType::Integer, 0, System::Data::ParameterDirection::
                        Input, false, (Byte)0, (Byte)0, "Номер п/п",
                        System::Data::DataRowVersion::Original, nullptr));
        Адаптер->UpdateCommand = Команда;
        Команда->Connection = Підключення;
        try
        {// Update повертає кількість змінених рядків
                int kol = Адаптер->Update(НабірДаних, "БД телефонів");
                MessageBox::Show("Оновлено " + kol + " записів", "Оновлення таблиці",
                        MessageBoxButtons::OK, MessageBoxIcon::Information);
        }
        catch (Exception ^ Ситуація)
        {
                MessageBox::Show(Ситуація->Message, "Оновлення таблиці",
                        MessageBoxButtons::OK, MessageBoxIcon::Error);
        }
}
};
}
```

Як видно з коду, ми маємо три процедури обробки подій: завантаження форми MyForm_Load, клацання на кнопці Читати з БД button1_Click і клацання на кнопці Зберегти в БД button2_Click. Щоб об'єкти класів DataSet, DataAdapter, Connection і Command було видно в цих трьох процедурах, оголошуємо ці об'єкти зовнішніми всередині класу MyForm.

При програмування читання з бази даних спочатку за допомогою SQL-запиту ми вибрали всі записи з таблиці (Select * From [БД телефонів]) і за допомогою об'єкта класу Adapter помістили в набір даних DataSet. А потім вказали об'єкт класу DataSet як джерело (DataSource) для сітки даних dataGridView1. Фрагмент роботи програми після читання з бази даних представлений на рис. 6.16.

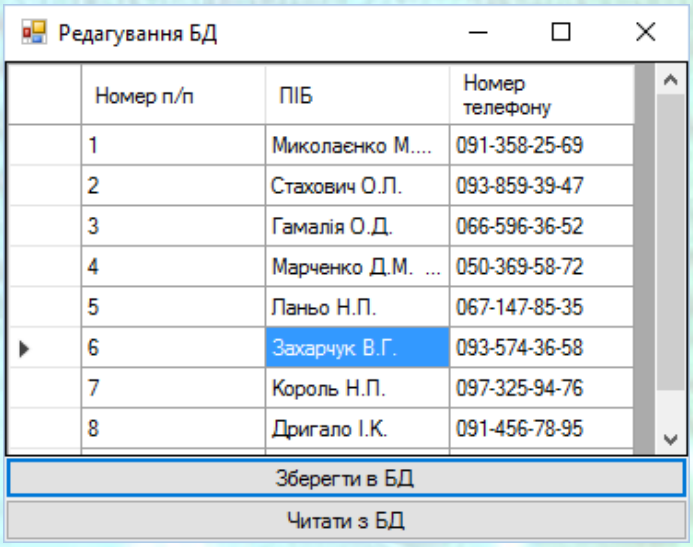

Рисунок 6.16 – Приклад роботи програми редагування таблиці даних

Для нас буде представляти інтерес програмування модифікації записів бази даних. Ця можливість реалізується при обробці події «клацання мишею на кнопці Зберегти в БД». Тут властивості CommandText присвоєно значення тексту SQL-запиту. В якості замінників параметрів використовуються знаки питання. В даному SQL-запиті мають місце три знаки питання. Їм відповідають три параметра, які повинні вказуватися суворо в порядку проходження знаків питання. Ці параметри задаємо з використанням методу Parameters->Add. Тут вказуємо ім'я поля (наприклад, Номер телефону), тип, довжину параметра і значення за замовчуванням. Зауважимо, що третій параметр (Номер п/п) задається як новий, оскільки він не повинен підлягати редагуванню з боку користувача, а буде встановлюватися автоматично, незалежно від дій користувача.

Далі в блоці try ... catch викликаємо безпосередньо метод Update, який повертає кількість (kol) оновлених записів. У разі невдалого оновлення обробляється виняткова ситуація Exception: об'єкт Exception забезпечує відповідне повідомлення про помилку.

6.8 Видалення записів з таблиці бази даних з використанням SQL-запиту і об'єкта класу Command

Можна також видаляти записи (рядки з таблиці БД), формуючи в програмному коді відповідний SQL-запит, який передається в об'єкт класу Command. Саме об'єкт Command забезпечує прив'язку SQL-виразу до з'єднання з базою даних. Напишемо найпростіший приклад такої програми.

Для вирішення поставленої задачі достатньо наступного програмного коду.

```
private: System::Void button3_Click(System::Object^  sender, System::EventArgs^  e) {
```

```
        // Створюємо об'єкт Connection і передаємо йому рядок підключення
        // Рядок підключення:
        auto Підключення = gcnew Data::OleDb::OleDbConnection(
                    "Data Source = D:\\Нова_БД.mdb; User ID = Admin;" +
                    " Provider = Microsoft.Jet.OLEDB.4.0;");
        Підключення->Open();
        // Створюємо об'єкт класу Command, передаючи йому SQL-команду
        auto Команда = gcnew Data::OleDb::OleDbCommand(
               "Delete * From [БД телефонів] Where ПІБ Like 'Ма%'", Підключення);
        // Виконання команди SQL
        int i = Команда->ExecuteNonQuery();
        // i - кількість видалених записів
        if (i> 0) MessageBox::Show(
               "Вилучено " + i.ToString() + " записів, які містять в полі
               ПІБ фрагмент 'Ма*'",
               "Видалення записів", MessageBoxButtons::OK, MessageBoxIcon::Information);
        if (i == 0) MessageBox::Show(
               "Запис, що містить в полі ПІБ фрагмент 'Ма*', не знайдено",
               "Видалення записів", MessageBoxButtons::OK, MessageBoxIcon::Information);
        Підключення->Close(); }
};
}
```

В даному випадку ми скористались програмою з попереднього прикладу, в яку додали об’єкт `button3` і програмний код розмістили у обробнику події – клацання на кнопці `button3_Click`. Кнопка `button3` була розміщена всередині об’єкту `panel1` та вирівняна за нижнім краєм, а також її властивості `Text` ми присвоїли значення `"Видалити записи"`. В результаті вікно програми набуло вигляду (рис. 6.17).

Тут при створенні об'єкта класу `OleDbCommand` заданий SQL-запит на видалення (`Delete`) всіх записів, що містить в полі ПІБ фрагмент тексту Ма*, причому малі та великі літери є рівнозначними, тобто будуть видалені записи, що містять Ма*, ма*, МА* та інші комбінації. Таким чином, пошук записів ведеться без урахування регістру (`case-insensitive search`).

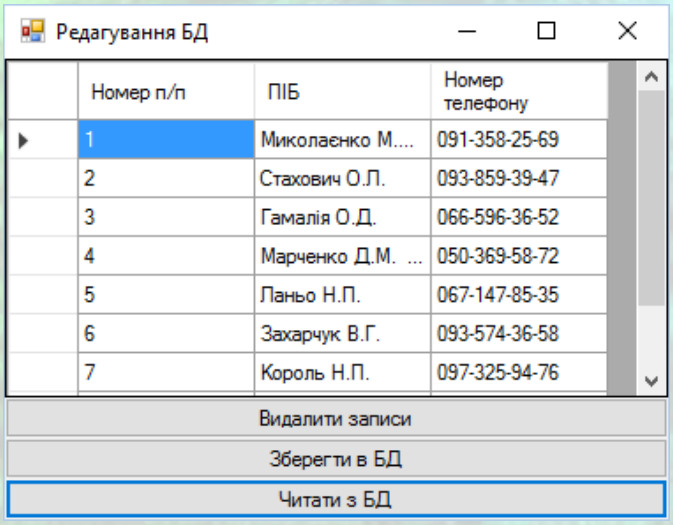

Рисунок 6.17 – Приклад роботи програми редагування таблиці даних з видаленням записів

Зауважимо, що тут для виконання команди SQL використаний метод `ExecuteNonQuery()`. Він повертає в змінну `i` кількість видалених записів (рис. 6.18).

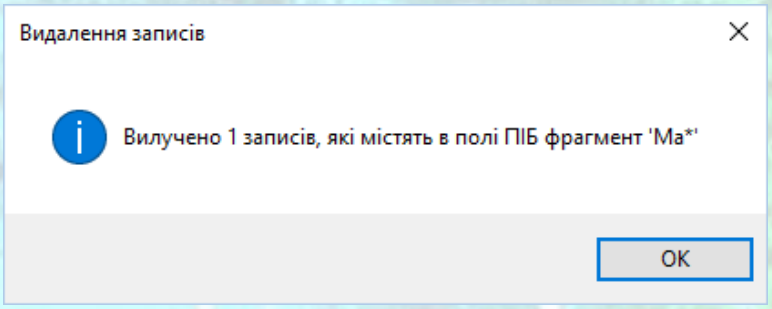

Рисунок 6.18 – Повідомлення про видалення записів з таблиці

Якщо `i = 0`, значить, записів з таким контекстом, не знайдено, і жоден запис не очищено.

В лістингу 6.1 наведений код програми, яка дозволяє переглядати, редагувати та видаляти записи з таблиці бази даних.

Програмний код

Лістинг 6.1

```
// Програма перегляду, редагування та видалення записів з таблиці БД
#pragma once
namespace WindowsForms {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;
	using namespace System::Data::OleDb;

	/// <summary>
	/// Сводка для MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
		}

	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}

	private: System::Windows::Forms::Panel^  panel1;
	private: System::Windows::Forms::Button^  button2;
	private: System::Windows::Forms::Button^  button1;
	private: System::Windows::Forms::Panel^  panel2;
	private: System::Windows::Forms::DataGridView^  dataGridView1;
	private: System::Windows::Forms::Button^  button3;
	protected:

	private:
		/// <summary>
		/// Обязательная переменная конструктора.
		/// </summary>
		System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
		/// <summary>
		/// Требуемый метод для поддержки конструктора — не изменяйте
		/// содержимое этого метода с помощью редактора кода.
```

```
		/// </summary>
		void InitializeComponent(void)
		{
			this->panel1 = (gcnew System::Windows::Forms::Panel());
			this->button2 = (gcnew System::Windows::Forms::Button());
			this->button1 = (gcnew System::Windows::Forms::Button());
			this->panel2 = (gcnew System::Windows::Forms::Panel());
			this->dataGridView1 = (gcnew System::Windows::Forms::DataGridView());
			this->button3 = (gcnew System::Windows::Forms::Button());
			this->panel1->SuspendLayout();
			this->panel2->SuspendLayout();
			(cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>dataGridView1))->BeginInit();
			this->SuspendLayout();
			//
			// panel1
			//
			this->panel1->Controls->Add(this->button3);
			this->panel1->Controls->Add(this->button2);
			this->panel1->Controls->Add(this->button1);
			this->panel1->Location = System::Drawing::Point(0, 161);
			this->panel1->Name = L"panel1";
			this->panel1->Size = System::Drawing::Size(284, 100);
			this->panel1->TabIndex = 3;
			//
			// button2
			//
			this->button2->Location = System::Drawing::Point(157, 74);
			this->button2->Name = L"button2";
			this->button2->Size = System::Drawing::Size(75, 23);
			this->button2->TabIndex = 4;
			this->button2->Text = L"button2";
			this->button2->UseVisualStyleBackColor = true;
			this->button2->Click += gcnew System::EventHandler(this,
&MyForm::button2_Click);
			//
			// button1
			//
			this->button1->Location = System::Drawing::Point(47, 74);
			this->button1->Name = L"button1";
			this->button1->Size = System::Drawing::Size(75, 23);
			this->button1->TabIndex = 3;
			this->button1->Text = L"button1";
			this->button1->UseVisualStyleBackColor = true;
			this->button1->Click += gcnew System::EventHandler(this,
&MyForm::button1_Click);
			//
			// panel2
			//
			this->panel2->Controls->Add(this->dataGridView1);
			this->panel2->Location = System::Drawing::Point(32, 26);
			this->panel2->Name = L"panel2";
			this->panel2->Size = System::Drawing::Size(200, 100);
			this->panel2->TabIndex = 4;
			//
			// dataGridView1
			//
			this->dataGridView1->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;
			this->dataGridView1->Location = System::Drawing::Point(-42, -28);
			this->dataGridView1->Name = L"dataGridView1";
```

```
                    this->dataGridView1->Size = System::Drawing::Size(284, 157);
                    this->dataGridView1->TabIndex = 1;
                    //
                    // button3
                    //
                    this->button3->Location = System::Drawing::Point(107, 45);
                    this->button3->Name = L"button3";
                    this->button3->Size = System::Drawing::Size(75, 23);
                    this->button3->TabIndex = 5;
                    this->button3->Text = L"button3";
                    this->button3->UseVisualStyleBackColor = true;
                    this->button3->Click += gcnew System::EventHandler(this,
&MyForm::button3_Click);
                    //
                    // MyForm
                    //
                    this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                    this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                    this->ClientSize = System::Drawing::Size(284, 261);
                    this->Controls->Add(this->panel2);
                    this->Controls->Add(this->panel1);
                    this->Name = L"MyForm";
                    this->Text = L"MyForm";
                    this->Load += gcnew System::EventHandler(this, &MyForm::MyForm_Load);
                    this->panel1->ResumeLayout(false);
                    this->panel2->ResumeLayout(false);
                    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>dataGridView1))->EndInit();
                    this->ResumeLayout(false);

                }
#pragma endregion

                // Програма оновлює записи (Update) в таблиці бази даних MS Access
                // ~ ~ ~ ~ ~ ~ ~ ~
                // Оголошуємо ці змінні поза всіма процедур, щоб
                // Вони були видні з будь-якої з процедур:
                DataSet ^ НабірДаних;
                OleDb::OleDbDataAdapter ^ Адаптер;
                OleDb::OleDbConnection ^ Підключення;
                OleDb::OleDbCommand ^ Команда;
private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
       panel1->Height = button1->Height*3;
       button3->Dock = System::Windows::Forms::DockStyle::Bottom;
       button2->Dock = System::Windows::Forms::DockStyle::Bottom;
       button1->Dock = System::Windows::Forms::DockStyle::Bottom;
       panel1->Dock = System::Windows::Forms::DockStyle::Bottom;
       panel2->Dock = System::Windows::Forms::DockStyle::Fill;
       dataGridView1->Dock = System::Windows::Forms::DockStyle::Fill;
       Text = "Редагування БД";
       НабірДаних = gcnew DataSet();
       Підключення = gcnew OleDb::
              OleDbConnection("Data Source=D:\\Нова_БД.mdb; User ID=" +
                     "Admin; Provider=Microsoft.Jet.OLEDB.4.0;");
       Команда = gcnew OleDb::OleDbCommand();
       button1->Text = "Читати з БД";
       button1->TabIndex = 0;
       button2->Text = "Зберегти в БД";
       button3->Text = "Видалити записи";
       }
```

```
private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e) {
	// Читати з БД:
	if (Підключення->State ==
		ConnectionState::Closed) Підключення->Open();
	Адаптер = gcnew OleDb::OleDbDataAdapter(
		"Select * From [БД телефонів]", Підключення);
	// Заповнюємо DataSet результатом SQL-запиту
	Адаптер->Fill(НабірДаних, "БД телефонів");
	// Вміст DataSet у вигляді рядка XML для відладки:
	String ^ РядокXML = НабірДаних->GetXml();
	// Вказати джерело даних для сітки даних:
	dataGridView1->DataSource = НабірДаних;
	// Вказуємо ім'я таблиці в наборі даних:
	dataGridView1->DataMember = "БД телефонів";
	Підключення->Close();
	}

private: System::Void button2_Click(System::Object^  sender, System::EventArgs^  e) {
	// Зберегти в базі даних
	Команда->CommandText = "UPDATE [БД телефонів] SET [Номер телефону] =?," +
		" ПІБ =? WHERE ([Номер п/п] =?)";
	// Ім'я, тип і довжина параметра
	Команда->Parameters->Add("Номер телефону",
		OleDb::OleDbType::VarWChar, 50, "Номер телефону");
	Команда->Parameters->Add(
		"ПІБ", OleDb::OleDbType::VarWChar, 50, "ПІБ");
	Команда->Parameters->Add
		(gcnew OleDb::OleDbParameter("Original_Номер_п_п",
			OleDb::OleDbType::Integer, 0, System::Data::ParameterDirection::
			Input, false, (Byte)0, (Byte)0, "Номер п/п",
			System::Data::DataRowVersion::Original, nullptr));
	Адаптер->UpdateCommand = Команда;
	Команда->Connection = Підключення;
	try
	{// Update повертає кількість змінених рядків
		int kol = Адаптер->Update(НабірДаних, "БД телефонів");
		MessageBox::Show("Оновлено " + kol + " записів", "Оновлення таблиці",
MessageBoxButtons::OK, MessageBoxIcon::Information);
	}
	catch (Exception ^ Ситуація)
	{
		MessageBox::Show(Ситуація->Message, "Оновлення таблиці",
MessageBoxButtons::OK, MessageBoxIcon::Error);
	}
	}

private: System::Void button3_Click(System::Object^  sender, System::EventArgs^  e) {
	// Створюємо об'єкт Connection і передаємо йому рядок підключення
	// Рядок підключення:
	auto Підключення = gcnew Data::OleDb::OleDbConnection(
			"Data Source = D:\\Нова_БД.mdb; User ID = Admin;" +
			" Provider = Microsoft.Jet.OLEDB.4.0;");
	Підключення->Open();
	// Створюємо об'єкт класу Command, передаючи йому SQL-команду
	auto Команда = gcnew Data::OleDb::OleDbCommand(
		"Delete * From [БД телефонів] Where ПІБ Like 'Ма%'", Підключення);
	// Виконання команди SQL
	int i = Команда->ExecuteNonQuery();
	// i - кількість видалених записів
	if (i> 0) MessageBox::Show(
		"Вилучено " + i.ToString() +
```

```
                " записів, які містять в полі ПІБ фрагмент 'Ма*'",
                "Видалення записів", MessageBoxButtons::OK, MessageBoxIcon::Information);
        if (i == 0) MessageBox::Show(
                "Запис, що містить в полі ПІБ фрагмент 'Ма*', не знайдено",
                "Видалення записів", MessageBoxButtons::OK, MessageBoxIcon::Information);
        Підключення->Close();
}
};
}
```

Завдання до комп'ютерного практикуму №6

Вимоги

1) Розробити структуру таблиці бази даних не меньше ніж з 6 полів, яка міститиме обов'язково цілочисельні значення, значення з плаваючою комою, текстові поля.
2) Створити базу даних у форматі `*.mdb` програмними засобами в середовищі CLR `Visual C++`.
3) Записати структуру розробленої таблиці даних в створену порожню базу даних MS `Access` за допомогою технології `ADO.NET`.
4) Створити інтерфейс програмного модуля з використанням об'єкту `TabControl` та кількома сторінками-вкладками – об'єктами `TabPage`.
5) На одній сторінці - об'єкті `TabPage` створити інтерфейс для керування базою даних з її відображенням в об'єкті `DataGridView`.
6) Додати до їншої сторінки `TabPage` об'єкт `Chart` для побудови діаграм.
7) Наповнити створену таблицю даних, з використанням розробленого програмного модуля, не менше ніж 30 неповторюваними записами.
8) Передбачити можливість формування користувачем не менше ніж трьох різних типів запитів на вибірку, в тому числі відбір записів за умовою та видалення записів за умовою.
9) Побудувати в об'єкті `Chart` діаграму на основі інформації з таблиці бази даних.

Контрольні питання

1) Що таке технологія ADO.NET?
2) Наведіть переваги технології ADO.NET?
3) Що таке постачальник даних .NET Framework? Які постачальники даних включає .NET Framework?
4) Як додати до проекту DLL-бібліотеки ADO?
5) Як програмно створити нову базу даних MS Access *.mdb?
6) Як записати структуру нової таблиці в базу даних MS Access за допомогою технології ADO.NET?
7) Як сформувати SQL-запит за допомогою екземпляру класу OleDbCommand?
8) Для чого використовується екземпляр класу OleDbConnection?
9) Для чого використовується метод ExecuteNonQuery()?
10) Як додати записи до таблиці бази даних?
11) Як перевірити виконання підключення до бази даних в режими проектування?
12) Як створити новий запит за допомогою Оглядача серверів?
13) Як вивести таблицю БД в об'єкт класу DataGridView за допомогою об'єктів класів Command та DataReader?
14) Як вивести таблицю БД в об'єкт класу DataGridView за допомогою об'єктів класів Command, Adapter та DataSet?
15) Як організувати редагування записів в таблиці бази даних?
16) Як видалити записи з таблиці бази даних за допомогою SQL-запиту?

Комп'ютерний практикум №7

Використання функцій зовнішніх програм

Мета: вивчити прийоми роботи з функціями зовнішніх програм. Ознайомитись з деякими бібліотеками зовнішніх програм, навчитися їх підключати до програмного модуля та використовувати функції зовнішніх бібліотек для більш ефективного вирішення поставлених задач.

Завдання: створити новий пустий проект CLR, додати в нього форму Windows Forms. Додати необхідні зовнішні бібліотеки. Перенести потрібні об'єкти з Панелі елементів для створення головного та контекстного меню і панелі інструментів. Додати до форми інші об'єкт, необхідні для створення інтерфейсу користувача для вирішення поставленої задачі. Створити об'єкти потрібних класів для реалізації поставленої задачі згідно отриманого варіанту завдання.

Загальні вимоги.

1) Створити проект Windows Forms.
2) Додати в нього необхідні елементи керування та програмний код для реалізації поставленої задачі згідно отриманого варіанту завдання
3) Програмний код має бути чітко структурований.
4) Імена об'єктів мають нести сенсові навантаження.
5) Програмний код має супроводжуватись коментарями в тексті програми.

Вимоги до виконання.

1) Створити новий пустий проект CLR в Visual C++ з використанням створеного шаблону проектів Windows Forms з ім'ям виду Прізвище_КПР7.
2) Додати у форму об'єкт головного меню класу MenuStrip.
3) Додати у форму об'єкт контекстного меню класу ContextMenuStrip.
4) Додати у форму об'єкт панелі інструментів класу ToolStrip.
5) Додати об'єкт DataGridView для відображення таблиці даних, в який будуть вводитись матриця коефіцієнтів та стовпчик вільних членів системи лінійних алгебраїчних рівнянь.
6) Додати інші необхідні об'єкти для створення програмного інтерфейсу та запрограмувати їх роботу.
7) Передбачити збереження табиці даних у зовнішньому файлі.
8) Передбачити можливість відкриття збереженого файлу таблиці в об'єкті DataGridView вікна додатку.
9) Підключити до проекту зовнішні бібліотеки для використання функцій MS Word та MS Excel.

10) Вирішити систему лінійних алгебраїчних рівнянь у матричній формі за допомогою функцій MS Excel з підключеної бібліотеки Microsoft Excel Object Library.
11) Результати розрахунку разом з системою рівнянь підготувати у вигляді звіту у MS Word за допомогою підключеної бібліотеки Microsoft Word Object Library.
12) Передбачити можливість збереження файлу MS Word з сформованим звітом за допомогою стандартного діалогового вікна збереження файлів SaveFileDialog з вікна програми.
13) Протягом роботи програми вікна MS Word та MS Excel не мають з'являтися, таким чином, щоб користувач і не здогадувався про використання бібліотек функцій зовнішніх програм.

Теоретичні відомості

7.1 Перевірка правопису засобами MS Word

Пакет програм Microsoft Office може бути сервером OLE-об'єктів, і його функції можуть використовуватися іншими додатками. Слід зазначити, що програмувати взаємодію програми на Visual C++ з різними офісними додатками (Word, Excel, Access, PowerPoint і т. д.), а також з AutoCAD і CorelDRAW зручно, оскільки в усі ці додатки вбудовано мову VBA (Visual Basic for Applications), в арсеналі якої знаходяться програмні об'єкти, назви і призначення яких багато в чому схожі з об'єктами, що використовуються в Visual C++/CLI (так само як і в Visual C#). Крім того, існує можливість запису макросу з подальшим переглядом відповідної VBA-програми.

Щоб скористатися можливостями MS Word потрібно до поточного проекту додати об'єктну бібліотеку (бібліотеку компонентів). Для цього правою кнопкою міші клацнемо по імені проекту у вікні Оглядача рішень (Solution explorer) і з контекстного меню оберемо Додати → Посилання... (рис. 7.1).

Або виділимо мишею у вікні Оглядача рішень (Solution explorer) ім'я проекту, а потім оберемо з меню Проект (Project) пункт меню Додати посилання... (Add Reference...) (рис. 7.2).

Потім, якщо на вашому комп'ютері встановлений MS Office 2003, то на вкладці COM двічі клацнемо по посиланню на бібліотеку Microsoft Word 11.0 Object Libary. Якщо встановлено MS Office 2007, то двічі клацнемо на посилання Microsoft Word 12.0 Object Library. Для MS Office 2016 обираємо посилання Microsoft Word 16.0 Object Library та натискаємо OK у вікні Додати посилання (Add Reference) (рис. 7.3).

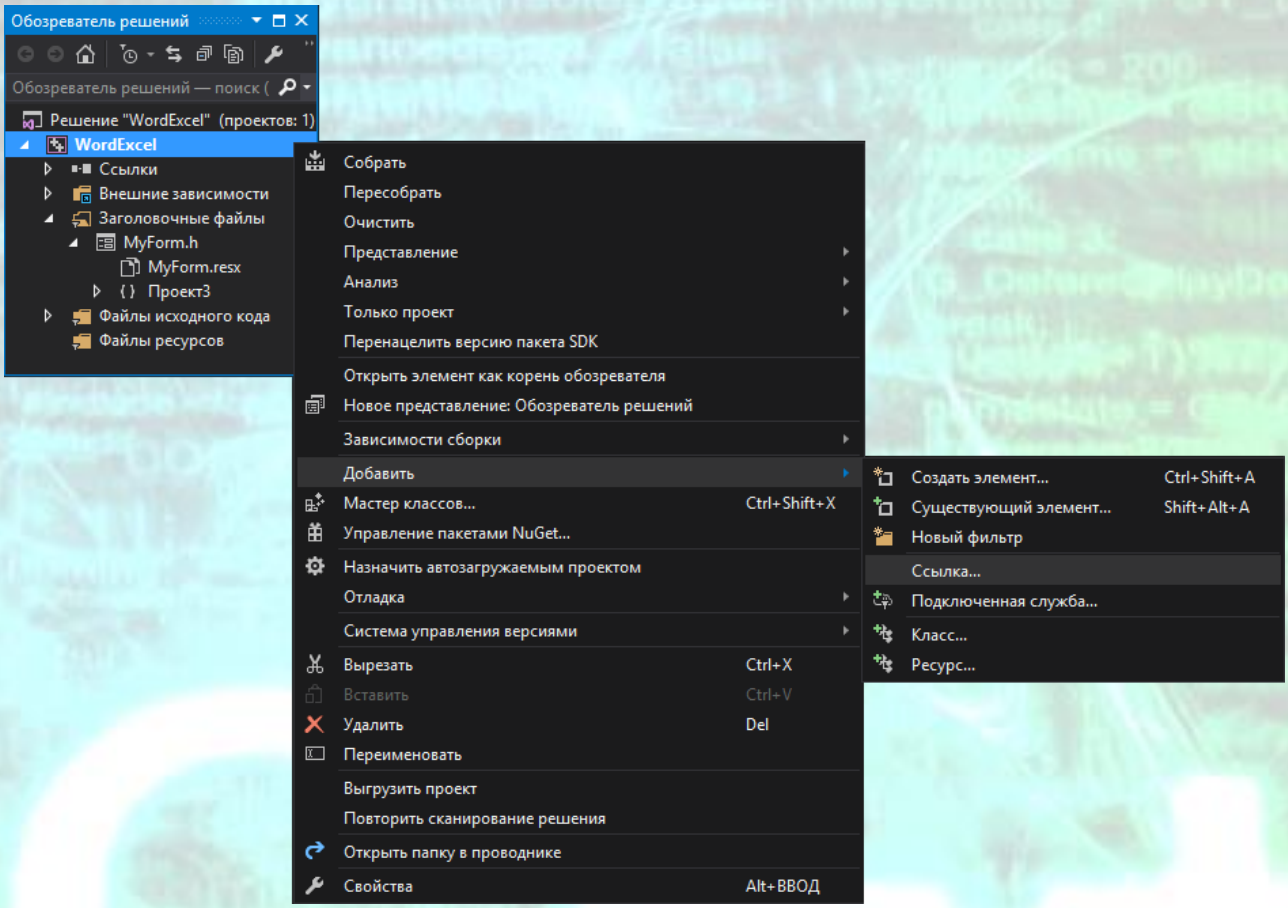

Рисунок 7.1 – Додавання посилання на зовнішню бібліотеку

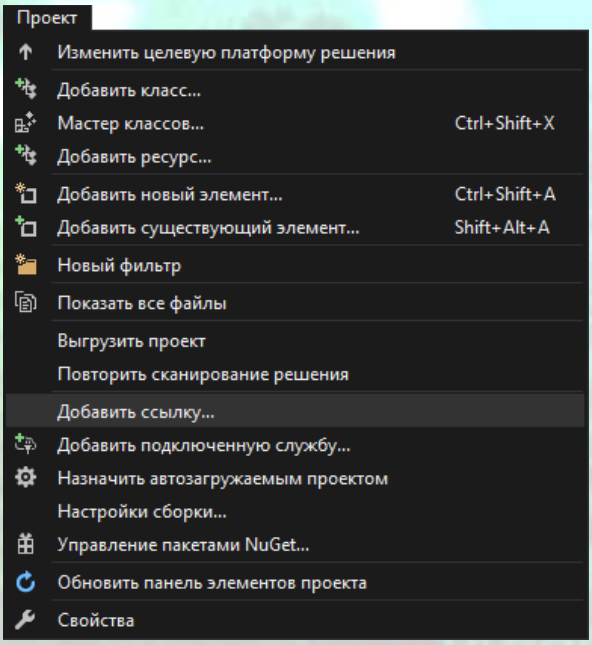

Рисунок 7.2 – Додавання посилання на зовнішню бібліотеку

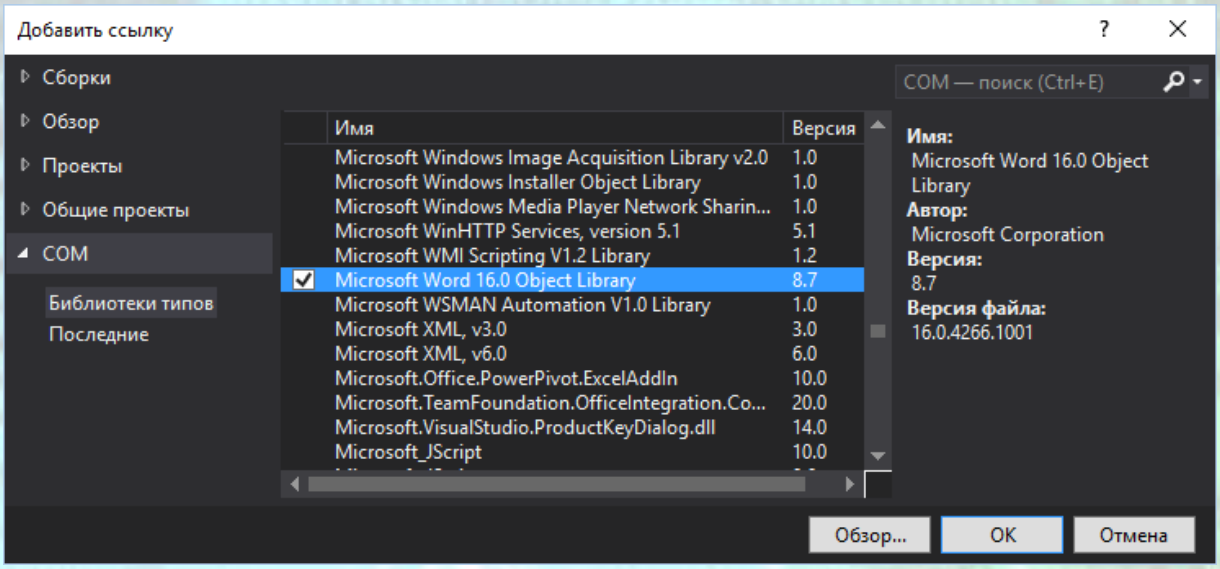

Рисунок 7.3 – Діалогове вікно додавання посилання на зовнішню бібліотеку

Ця об'єктна бібліотека відповідає файлу, розташованому за адресою: `C:\Program Files\Microsoft Office\Office16\MSWORD.OLB` (`C:\Program Files\Microsoft Office\OFFICE11\MSWORD.OLB` для MS Office 2003 або `C:\Program Files\Microsoft Office\OFFICE12\MSWORD.OLB` для MS Office 2007).

Тепер переконаємося в тому, що дане посилання благополучно встановлене. Для цього у вікні Оглядача рішень (Solution explorer) нашого проекту у групі Посилання (References) знайдемо додане в наш проект обране посилання `Interop.Microsoft.Office.Interop.Word.8.7` (рис. 7.4).

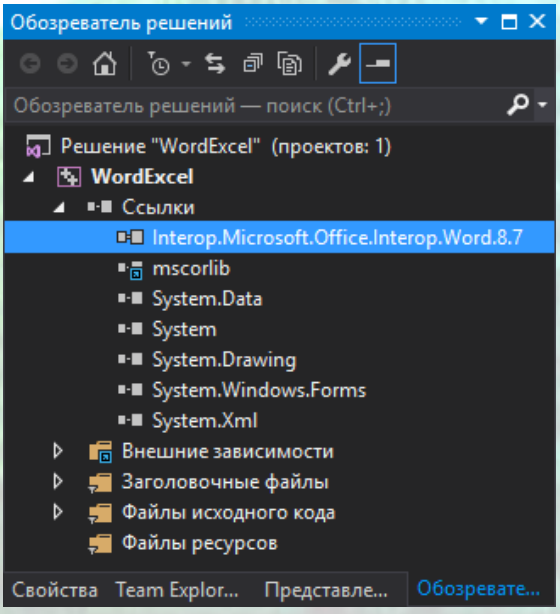

Рисунок 7.4 – Вікно Оглядача рішень з доданою зовнішньою бібліотекою Microsoft Word 16.0 Object Library

Також у папці проекту з'явиться папка Interop в якій буде розміщено файл бібліотеки Interop.Microsoft.Office.Interop.Word.8.7.dll. Таким чином, ми підключили бібліотеку об'єктів MS Word.

Тепер можна використовувати функції MS Word в нашій програмі для виконання поставлених задач.

Приклад програмної реалізації

Приклад створення додатку з перевіркою правопису засобами MS Word.

Створимо програму, яка дозволяє користувачеві ввести будь-які слова, речення в текстове поле і після натиснення відповідної кнопки перевірити орфографію введеного тексту. Для безпосередньої перевірки орфографії скористаємося функцією Checkspelling об'єктної бібліотеки MS Word.

Для вирішення цієї задачі запустимо Visual Studio і в вікні Створення проекту (New Project) виберемо в середовищі CLR вузла Visual C++ додаток шаблону Windows Forms. Перенесемо з Панелі елементів (Toolbox) в проектовану форму елемент управління Panel і в ньому розмістимо елемент RichTextBox. Також на формі розмістимо елемент Button.

Далі введемо програмний код, представлений в лістингу нижче.

```
#pragma endregion

private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
        // У пункті меню Project виберемо команду Add Reference.
        // Потім, якщо на вашому комп'ютері встановлений MS Office 2007,
        // То на вкладці COM двічі клацнемо по посиланню
        // На бібліотеку Microsoft Word 12.0 Object Libary.
        richTextBox1->Clear();
        button1->Text = "Перевірка орфографії";
        richTextBox1->TabIndex = 0;
        button1->TabIndex = 1;
        button1->Dock = System::Windows::Forms::DockStyle::Bottom;
        panel1->Dock = System::Windows::Forms::DockStyle::Fill;
        richTextBox1->Dock = System::Windows::Forms::DockStyle::Fill;
        Text = "Орфографія";
}

private: System::Void button1_Click_1(System::Object^  sender, System::EventArgs^  e) {
        auto Ворд1 = gcnew Microsoft::Office::Interop::Word::Application();
        // Ворд1-> Visible = false;
        System::Object ^ t = Type::Missing;
        // Відкриваємо новий порожній документ:
        auto Документ = Ворд1->Documents->Add(t, t, t, t);
        // Вводимо в порожній документ текст з текстового поля:
        Документ->Words->First->InsertBefore(richTextBox1->Text);
        // Перевірка орфографії:
        Документ->CheckSpelling(t, t, t, t, t, t, t, t, t, t, t, t);
        // Отримуємо виправлений текст:
        String ^ ВиправленнийТекст = Документ->Content-> default;
        // У Visual Basic і C# буде працювати так:
```

```
        // String^ ВиправленнийТекст = документ->Content->Text;
        // Повертаємо в текстове поле виправлений текст:
        richTextBox1->Text = ВиправленнийТекст->Replace("\r", "");
        System::Object ^ tt = false;
        // Або = tt Microsoft::Office::Interop::Word::WdSaveOptions::wdDoNotSaveChanges;
        Ворд1->Documents->Close(tt, t, t);
        // Закрити документ Word без збереження:
        Ворд1->Quit(tt, t, t);
        Ворд1 = nullptr;
}
};
}
```

Як видно з тексту програми, при обробці події завантаження форми MyForm_Load очищається текстове поле, ініціалізується назва кнопки Перевірка орфографії та заголовок вікна Орфографія, виконується вирівнювання елементів інтерфейсу в межах форми, а також за допомогою властивості TabIndex задається порядок переключення фокусу між об'єктами форми за допомогою клавіші Tab.

При обробці події клацання по кнопці Button1_Click створюється новий об'єкт Ворд1 класу Word::Application, і командою Documents->Add відкривається новий документ. Тут ми використовували поле Type::Missing для отримання значення параметра методу за замовчуванням. Далі весь введений користувачем текст копіюється в цей документ (команда InsertBefore). Потім відбувається безпосередня перевірка орфографії методом Checkspelling. Якщо текст перевірений, то правильний варіант написання слова знаходиться тепер в одноній з властивостей об'єкта класу Word::Document, а саме у властивості Content->Text. Якби ми писали програму на Visual Basic або на C#, то ми б до цієї властивості так і звернулися: Content->Text. Однак транслятор Visual C++ вимагає, щоб до цієї властивості ми звернулися, використовуючи ключове слово default. Тепер виправлений текст повертаємо назад в текстове поле. Далі документ MS Word закриваємо без збереження змін wdDoNotSaveChanges, тому при роботі програми ми цей документ навіть не помічаємо.

Тепер перевіримо, як працює написана нами програма. Запустимо її, натиснувши на функціональну клавішу F5, і в текстове поле введемо яке-небудь слово з помилкою. Після клацання на кнопці Перевірка орфографії отримаємо діалогове вікно, подібне представленому на рис. 7.5.

Виберемо правильний варіант написання слова і клацнемо на кнопці Замінити, при цьому діалогове вікно закриється, а в нашому текстовому полі на формі з'явиться виправлене слово.

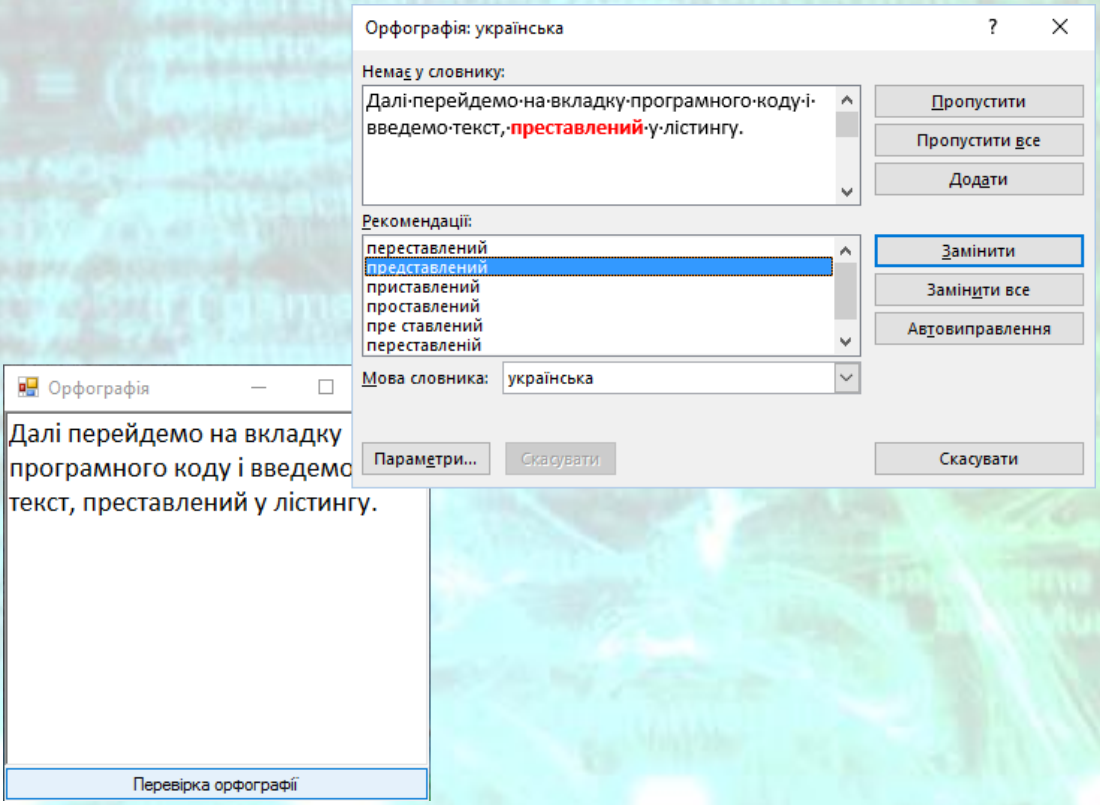

Рисунок 7.5 – Вікно програми з діалоговим вікном Орфографія

У наступному лістингу наведемо інший, більш короткий варіант вирішення цього завдання з використанням об'єкта класу `Selection`.

```
#pragma endregion

private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
        // У пункті меню Project виберемо команду Add Reference.
        // Потім, якщо на вашому комп'ютері встановлений MS Office 2007,
        // То на вкладці COM двічі клацнемо по посиланню
        // На бібліотеку Microsoft Word 12.0 Object Libary.
        richTextBox1->Clear();
        button1->Text = "Перевірка орфографії";
        richTextBox1->TabIndex = 0;
        button1->TabIndex = 1;
        button1->Dock = System::Windows::Forms::DockStyle::Bottom;
        panel1->Dock = System::Windows::Forms::DockStyle::Fill;
        richTextBox1->Dock = System::Windows::Forms::DockStyle::Fill;
        Text = "Орфографія";
}

private: System::Void button1_Click_1(System::Object^  sender, System::EventArgs^  e) {
        auto Ворд1 = gcnew Microsoft::Office::Interop::Word::Application();
        // Ворд1-> Visible = false;
        // Змінна з "порожнім" значенням:
        System::Object  ^ t = Type::Missing;
        // Відкриваємо новий порожній документ MS Word:
        Ворд1->Documents->Add(t, t, t, t);
        // Копіюємо вміст текстового вікна в документ
        Ворд1->Selection->default = richTextBox1->Text;
        // Для VB і C # було б: Ворд1->Selection->Text
```

```
        // Безпосередня перевірка орфографії:
        Ворд1->ActiveDocument->CheckSpelling(t, t, t, t, t, t, t, t, t, t, t, t);
        // Копіюємо результат назад в текстове поле
        richTextBox1->Text = Ворд1->Selection->default;
        Object ^ tt = false;
        Ворд1->Documents->Close(tt, t, t);
        // Закрити документ Word без збереження:
        Ворд1->Quit(tt, t, t);
        Ворд1 = nullptr;
}
};
}
```

У цьому програмному коді аналогічно створюємо новий об'єкт Ворд1 класу Word::Application і, використовуючи метод Add, відкриваємо документ MS Word. Далі скористаємося об'єктом Ворд1->Selection, він оперує з поточною позицією «невидимого» курсору, наприклад, може подати команду введення тексту в документ MS Word (метод TypeText). Ми використовуємо властивість Text цього об'єкта для копіювання вмісту текстового вікна в документ MS Word. Відповідно доступ до цієї властивості має бути реалізований за допомогою операторoa: Ворд1->Selection->Text. Так ми б і вчинили, якби писали програму на VB або C#. Однак транслятор C++ вимагає, щоб до цієї властивості ми звернулися, використовуючи ключове слово default. Далі відбувається безпосередня перевірка орфографії аналогічно попередньому варіанту вирішення цього завдання. Повний текст даного програмного модуля наведений в лістингу 7.1.

Приклад програмної реалізації

Приклад створення таблиці в MS Word.

Розглянемо спосіб створення таблиці з використанням функції MS Word. Отже, запускаємо Visual Studio і в вікні Створення проекту (New Project) виберемо в середовищі CLR вузла Visual C++ додаток шаблону Windows Forms для Visual C++. Далі до поточного проекту додамо об'єктну бібліотеку MS Word. Для цього в меню Проект (Project) вкажемо команду Додати посилання… (Add Reference…) і на вкладці COM двічі клацнемо по посиланню на бібліотеку Microsoft Word 16.0 Object Library (або інша версія MS Word, наприклад 12.0 Object Library). На екранну форму перенесемо командну кнопку Button, щоб робота програми виглядала більш виразно. Тобто саме після клацання на кнопці буде формуватися таблиця і викликатися MS Word для її відображення. Далі введемо програмний код, представлений в лістингу нижче.

```
#pragma endregion

        private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
```

```
            // У меню Project вкажемо команду Add Reference і на
            // Вкладці COM двічі клацнемо по посиланню на
            // Бібліотеку Microsoft Word 16.0 Object Library
            button1->Text = "Пуск"; this->Text = "Побудова таблиці";
        }

        private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e) {
            // Ініціалізуємо два рядкових масиви:
            array <String ^> ^ Імена = { "Андрій (роб)", "Світла (моб)", "Микола (дом)",
                "Кафедра (роб)", "Олександр Степанович", "Сергій (дом)",
                "Тетяна Петрівна", "Таксі", "Кінотеатр" };
            array <String ^> ^ Телефон = { "274-88-17", "+38 (067) 7030356", "22-345-72",
                "204-82-12", "223-67-67 доп 32-67", "570-38-76",
                "201-72-23", "555", "216-40-22" };
            // Створюємо новий екземпляр класу Word::Application:
            auto Ворд1 = gcnew Microsoft::Office::Interop::Word::Application();
            Ворд1->Visible = true;
            // Змінна з "порожнім" значенням:
            System::Object ^ t = Type::Missing;
            // Відкриваємо новий документ MS Word:
            auto Документ = Ворд1->Documents->Add(t, t, t, t);
            // Вводимо текст в документ MS WORD з поточної позиції:
            Ворд1->Selection->TypeText("ТАБЛИЦЯ ТЕЛЕФОНІВ");
            // Параметр, який вказує чи показувати межі комірок:
            System::Object ^ t1 = Microsoft::Office::Interop::
                Word::WdDefaultTableBehavior::wdWord9TableBehavior;
            // Параметр, який вказує чи буде додаток Word автоматично
            // змінювати розмір комірок у таблиці для підгонки вмісту:
            System::Object ^ t2 =
                Microsoft::Office::Interop::Word::WdAutoFitBehavior::wdAutoFitContent;
            // Створюємо таблицю з 9 рядків і 2 стовпців:
            Ворд1->ActiveDocument->Tables->Add(Ворд1->Selection->Range, 9, 2, t1, t2);
            // Заповнювати комірки таблиці можна так:
            for (int i = 1; i <= 9; i++)
            {
                Ворд1->ActiveDocument->Tables[1]->Cell(i, 1)->
                    default->InsertAfter(Імена[i - 1]);
                Ворд1->ActiveDocument->Tables[1]->Cell(i, 2)->
                    default->InsertAfter(Телефон[i - 1]);
                //Програмуючи на C# ми написали б:
                //Ворд1.ActiveDocument.Tables[1].Cell(i,2).Range.InsertAfter(Tel[i-1]);
            }
            // Призначаємо одиниці вимірювання в документі додатка MS Word:
            Object ^ t3 = Microsoft::Office::Interop::Word::WdUnits::wdLine;
            // Параметр, який вказує на дев'ятий рядок у документі MS Word:
            Object ^ рядок9 = 9;
            // Перевести поточну позицію (Selection) за межі таблиці,
            // (В дев'ятий рядок), щоб тут вивести будь-якій текст:
            Ворд1->Selection->MoveDown(t3, рядок9, t);
            // І тут друкуємо наступний текст:
            Ворд1->Selection->TypeText("Який-небудь текст після таблиці");
            // Зберігати документ немає сенсу, але це вирішить користувач:
            // або можна задати безпосередньо шлях та ім'я файлу
            Object ^ ім’яФайла = "D:\\Таблиця.docx";
            // і одразу зберегти
            Ворд1->ActiveDocument->SaveAs(ім’яФайла, t, t, t, t, t, t, t, t, t,
                t, t, t, t, t, t);
        }
        };
}
```

Дані знаходяться в двох масивах: Імена[] і Телефон[]. Ми створюємо екземпляр об'єкта Word::Application і відкриваємо новий документ Document::Add. Тут ми використовували поле Type::Missing для отримання значення параметра методу за замовчуванням. Потім демонструємо, як можна додавати будь-які тексти в новий документ з C++ - програми. Наприклад, ми вводимо в активний документ текст «ТАБЛИЦЯ ТЕЛЕФОНІВ», використовуючи об'єкт Selection, що визначає поточну позицію «невидимого» курсору і містить методи для введення в документ MS Word.

Потім ми створюємо таблицю, що складається з дев'яти рядків і двох стовпців, причому ширина стовпців буде регулюватися в залежності від вмісту комірок (wdAutoFitContent). Потім в циклі заповнюємо елементи таблиці і виводимо курсор (Selection) за межі таблиці, щоб написати будь-який текст.

Після запуску цієї програми прямо на наших очах в редакторі MS Word сформується таблиця (рис. 7.6), яку при бажанні можна редагувати, зберігати і роздруковувати на принтері.

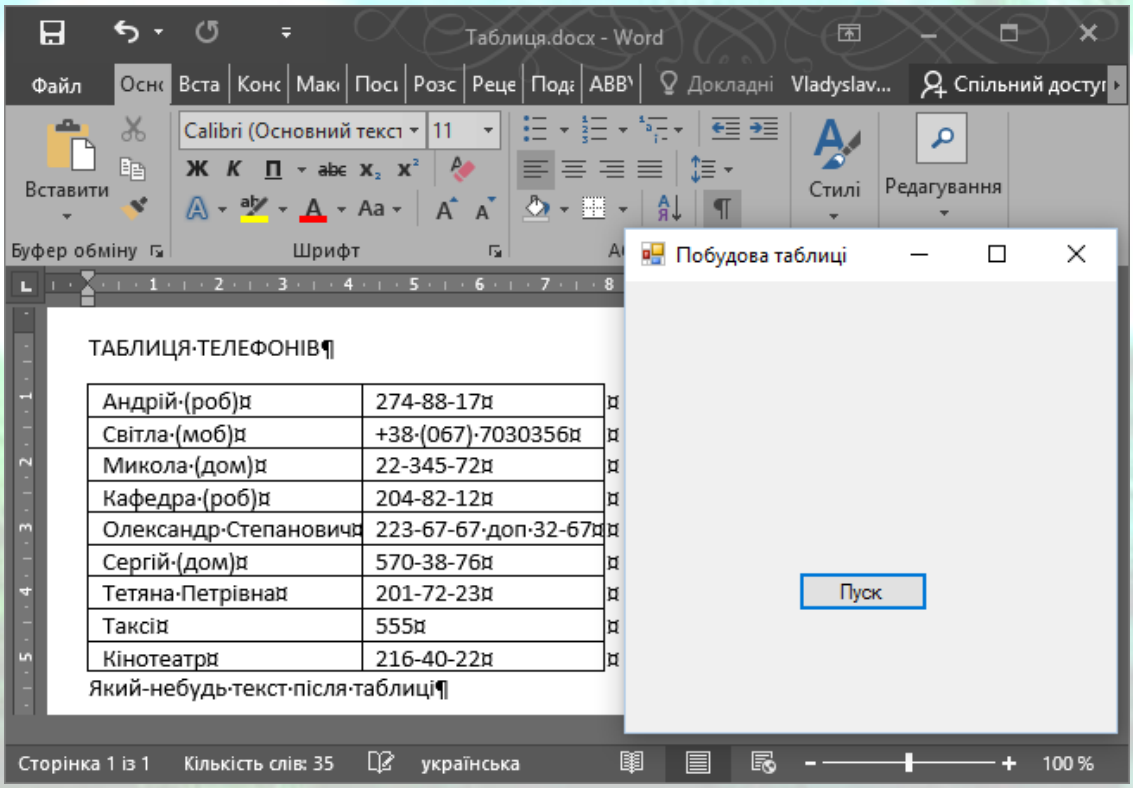

Рисунок 7.6 – Формування таблиці в MS Word з додатку C++/CLI

Окрім того, можна самостійно з додатку C++/CLI зберегти створений документ MS Word, якщо задати шлях збереження та ім'я файлу. Наприклад, наступним чином

```
// Задаємо безпосередньо шлях та ім'я файлу
Object ^ ім'яФайла = "D:\\Таблиця.docx";
```

```
// і одразу зберегти
Ворд1->ActiveDocument->SaveAs(ім'яФайла, t, t, t, t, t, t, t, t, t, t, t, t, t, t, t);
```

За допомогою інструментів роботи з MS Word дуже зручно формувати звіти роботи програми у зв'язку з широкими можливостями текстового редактору MS Word по підготовці, форматуванню та оформленню текстових документів.

Повний текст даного програмного модуля наведений в лістингу 7.2.

7.2 Використання функцій MS Excel

Розглянемо ще один приклад звернення до функцій MS Excel з програми на Visual C++. Припустимо, ми взяли кредит на покупку квартири 38 тис. доларів під 10% річних, на термін 20 років. Потрібно дізнатися суму, яку ми змушені будемо платити щомісяця. В україномовному MS Excel, як і в англомовному, для подібних розрахунків є функція PMT() (в російськомовній версії функція ПЛТ()), на вхід якої слід подати місячну процентну ставку (тобто в нашому випадку 0,10/12), термін погашення кредиту в місяцях (240 місяців) і розмір кредиту ($ 38 тис.). Аналогом функції PMT() є функція (метод) Pmt() класу WorksheetFunction, яка має такі ж аргументи. Список всіх методів (функцій) об'єкта WorksheetFunction з описом аргументів можна знайти за адресою: https://msdn.microsoft.com/en-us/library/bb225774.aspx .

Для програмування звернень до цих функцій з програми, створеної в Visual Studio, важливо знайти відповідність функцій MS Excel і їх аналогів в об'єкті WorksheetFunction для налагодження на тестових прикладах (для англомовних та україномовних версій Excel про це можна не турбуватись, бо функції мають однакові назви).

Запустимо Visual Studio і в вікні Створення проекту (New Project) виберемо в середовищі CLR вузла Visual C++ додаток шаблону Windows Forms для C++. У проектовану екранну форму з Панелі елементів (Toolbox) перенесемо три мітки, три текстових поля (для введення трьох перерахованих вище аргументів функції Pmt()) і кнопку. У поточний проект підключаємо бібліотеку об'єктів MS Excel. Для цього в меню Проект (Project) виберемо команду Додати посилання... (Add Reference...), потім на вкладці COM двічі клацнемо на посиланні Microsoft Excel 16.0 Object Library (рис. 7.7).

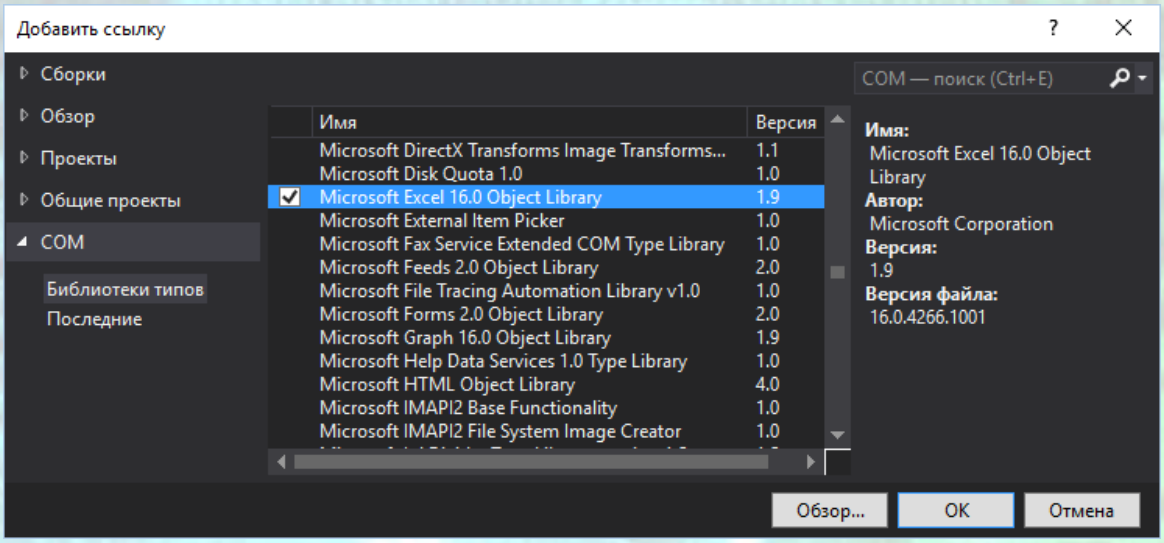

Рисунок 7.7 – Додавання бібліотеки

Для перевірки успішності додавання бібліотеки подивимось на групу Посилання (Reference) у вікні Оглядача рішень (Solution Explorer) нашого проекту. Там має з'явитись бібліотека Interop.Microsoft.Office.Interop.Excel.1.9 (рис. 7.8).

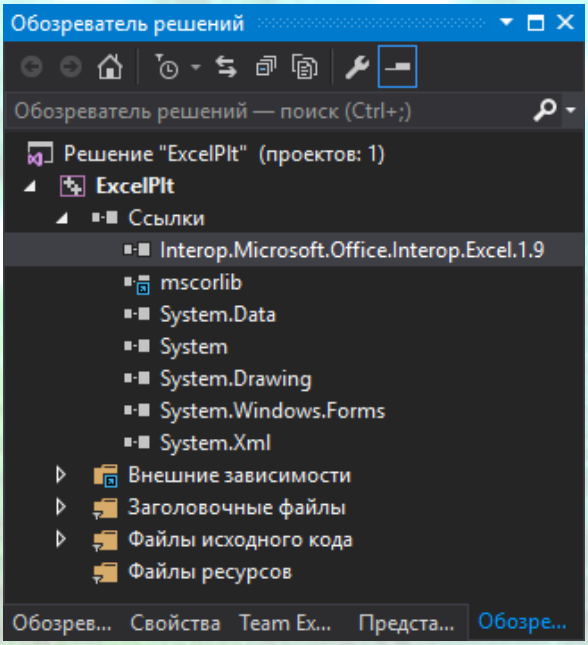

Рисунок 7.8 – Оглядач рішень з доданою бібліотекою Microsoft Excel 16.0 Object Library

Тепер можна перейти до програмного коду, наведеного в лістингу нижче.

```
#pragma endregion

private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
             this->Text = "Розрахунок щомісячних платежів";
             label1->Text = "Річна ставка у %";
```

```
            label2->Text = "Термін в місяцях";
            label3->Text = "Розмір кредиту";
            textBox1->Clear();
            textBox2->Clear();
            textBox3->Clear();
            button1->Text = "Розрахунок";
        }

    private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e) {
        try
        {
            auto XL = gcnew Microsoft::Office::Interop::Excel::Application();
            // Змінна з "порожнім" значенням:
            auto t = Type::Missing;
            // Отримуємо розмір місячного платежу:
            double pay = XL->WorksheetFunction->Pmt(
                (Convert::ToDouble(textBox1->Text))/1200,
                Convert::ToDouble(textBox2->Text),
                Convert::ToDouble(textBox3->Text), t, t);
            // АБО, якщо використовувати функцію Pmt()
            // з Microsoft VisualBasic:
            // Double FV = 0;
            // Microsoft::VisualBasic::DueDate dt =
            // Microsoft::VisualBasic::DueDate::EndOfPeriod;
            // Double pay = Microsoft::VisualBasic::Financial::Pmt(
            // (Convert::ToDouble(textBox1->Text))/1200,
            // Convert::ToDouble(textBox2->Text),
            // Convert::ToDouble(textBox3->Text), FV, dt);
            auto Рядок = String::Format(
                "Кожен місяць слід платити {0:$ #.##} доларів",
                Math::Abs(pay));
            MessageBox::Show(Рядок);
            XL->Quit();
        }
        catch (Exception ^ Ситуація)
        {
            MessageBox::Show(Ситуація->Message, "Помилка",
                MessageBoxButtons::OK, MessageBoxIcon::Exclamation);
        }
}
};
}
```

Як видно з тексту програми, при обробці події завантаження форми очищаються (Clear) текстові поля, а також підписуються назви цих полів за допомогою міток label1 - label3 і присвоюється назва кнопці Button1.

При обробці події клацання на кнопці Розрахунок створюється об'єкт Excel::Application. Об'єкт Excel::Application забезпечує доступ до функцій MS Excel, зокрема до функції Pmt(). На вхід функції Pmt() подаємо значення текстових полів, конвертованих з рядкового типу в тип Double. При цьому перший аргумент функції переводимо з річної процентної ставки в місячну ставку в сотих частках одиниці, тому ділимо на 1200. На вхід функції також доводиться подати ще два параметри, які є необов'язковими. Тут ми використовували поле Type::Missing для отримання значення параметра

методу за замовчуванням. На виході функції `Pmt()` отримуємо розмір місячного платежу, який виводимо, використовуючи функцію `MessageBox::Show`.

Звернення до функцій `MS Excel` оформляємо в блоці `try ... catch` для обробки виняткових ситуацій (`Exception`). Зауважимо, що в даній програмі ми не передбачили діагностику обов'язкового заповнення всіх полів, а також діагностику введення тільки числових даних, щоб не перевантажувати текст програми численними очевидними подробицями.

У даній програмі ми кілька змінних оголосили як `auto`, це означає, що тип відповідної змінної виводиться з виразу ініціалізації (як оголошення `var` в C#).

Інтерфейс програми показаний на рис. 7.9.

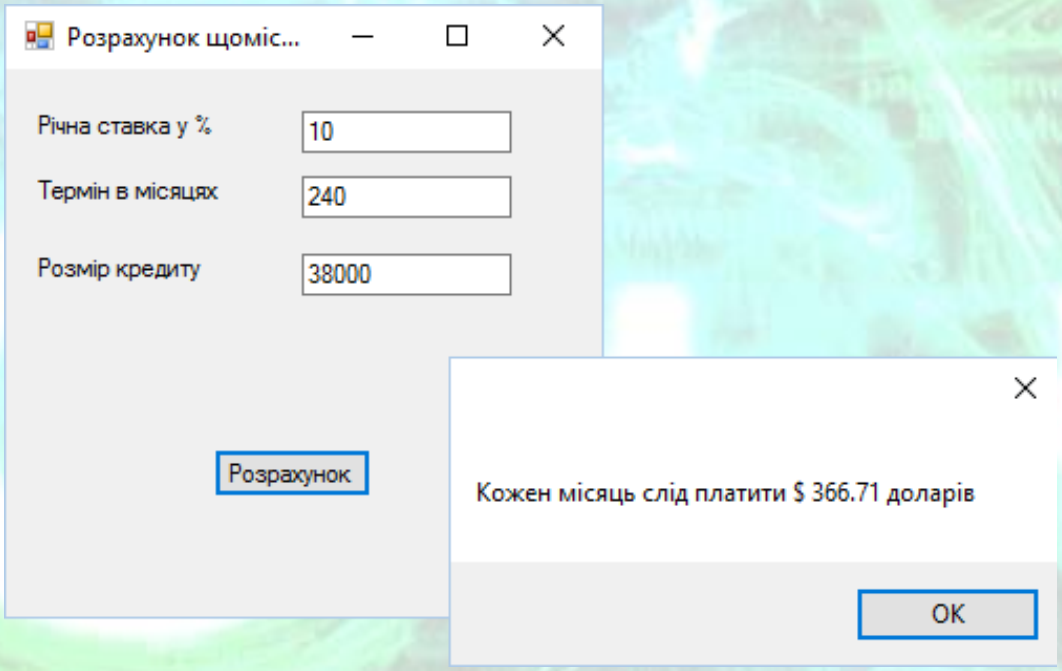

Рисунок 7.9 – Результат роботи програми з використанням фінансових функцій `MS Excel`

В даному підрозділі на простому прикладі ми розглянули, як легко підключитися до бібліотеки об'єктів `MS Excel` і користуватися її функціями. Однак функція `Pmt()` є також в середовищі `Visual Studio` в просторі імен `Microsoft::VisualBasic::Financial` точно з такими ж параметрами. Для звернення до цієї функції треба було б підключення до `Visual Basic`. Тобто слід було б в проект додати посилання на бібліотеку `Microsoft.VisualBasic.dll`. Для цього в пункті меню `Проект (Project)` треба вибрати команду `Додати посилання... (Add Reference...)`, а на вкладці `.NET` двічі клацнути на посиланні `Microsoft.VisualBasic`. В цьому випадку можна було б звертатися до функції `Pmt()`, як це представлено в коментарі. Однак в даному прикладі показана принципова можливість роботи з функціями `MS Excel` з C++ - програми.

Повний текст даного програмного модуля наведений в лістингу 7.3.

7.3 Рішення системи рівнянь за допомогою функцій MS Excel

Використовуючи функції MS Excel, в своїй програмі, створеній в Visual C++, можна вирішувати і більш серйозні завдання. Наприклад, розглянемо, як вирішити систему лінійних алгебраїчних рівнянь.

Вирішувати систему лінійних алгебраїчних рівнянь будемо в матричній формі. Система лінійних алгебраїчних рівнянь виду

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases}$$

в матричній формі записується як $A \cdot X = B$, де A – основна матриця системи, або матриця коефіцієнтів

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix},$$

X – матриця-стовпчик невідомих змінних

$$X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix},$$

B – матриця-стовпчик вільних членів

$$B = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}.$$

Рішення системи лінійних алгебраїчних рівнянь матричним методом визначається за залежністю:

$$X = A^{-1} \times B,$$

де A^{-1} – зворотна матриця коефіцієнтів при невідомих.

Для знаходження зворотної матриці в MS Excel існує функція MINVERSE() (в російськомовній версії МОБР()), аналогічна функції MInverse() в об'єкті WorksheetFunction з бібліотеки об'єктів Microsoft Excel.

Для множення зворотної матриці на вектор вільних членів є відповідно функції MMULT() (в російськомовній версії МПРОИЗ()) та MMult(). Такі функції відсутні в Visual Studio, і в даному випадку ми отримуємо реальний позитивний ефект від підключення до функцій MS Excel.

Розглянемо приклад, в якому знайдемо рішення для наступної системи лінійних алгебраїчних рівнянь:

$$x_1 + 2x_2 + x_3 = -1,$$

$$3x_1 - x_2 - x_3 = -1,$$
$$-2x_1 + 2x_2 + 3x_3 = 5$$

Приклад програмної реалізації

Приклад рішення системи лінійних рівнянь з використанням функцій MS Excel.

Для вирішення цієї задачі запустимо Visual Studio і в вікні Створення проекту (New Project) виберемо в середовищі CLR вузла Visual C++ додаток шаблону Windows Forms для C++. У проектовану екранну форму з Панелі елементів (Toolbox) перенесемо три мітки, в які запишемо наші рівняння, і кнопку. У поточний проект підключаємо бібліотеку об'єктів MS Excel. Для цього в меню Проект (Project) виберемо команду Додати посилання... (Add Reference...), потім на вкладці COM двічі клацнемо на посиланні Microsoft Excel 16.0 Object Library (рис. 7.7).

Програму побудуємо таким чином, що в обробнику події завантаження форми MyForm_Load запишемо наші алгебраїчні рівняння в об'єкти Label та додамо напис на кнопці. Розрахунок безпосередньо будемо проводити у обробнику події натискання кнопки button1_Click, тому в цьому обробнику події і задамо (ініціалізуємо) пряму матрицю у вигляді двовимірного масиву та рядок вільних членів у вигляді одновимірного масиву.

```
#pragma endregion

	private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e) {
		// Вирішуємо систему
		// x1 + 2*x2 + x3 = -1
		// 3*x1 - x2 - x3 = -1
		// -2*x1 + 2*x2 + 3*x3 = 5
		// Для цієї системи пряма матриця буде мати вигляд:
		// array<double,2> ^ A = gcnew array<double,2>(n, n);
		array<double, 2> ^ A = { { 1, 2, 1 },
                                         { 3, -1, -1 },
                                         { -2, 2, 3 } };
		// Рядок вільних членів:
		// array<double> ^ B = gcnew array<double>(n);
		// Вільні члени:
		array<double> ^ B = { -1, -1, 5 };
		// Створення екземпляра класу Excel::Application:
		auto XL1 = gcnew Microsoft::Office::Interop::Excel::Application();
		// Обчислення детермінанта матриці A
		double визначник_A = XL1->Application->WorksheetFunction->MDeterm(A);
		// Якщо визначник_A != 0, то вихід з процедури:
		if (визначник_A == 0)
		{
			MessageBox::Show("Система не має рішення, оскільки\n" +
				"визначник дорівнює нулю", "Немає рішення",
				MessageBoxButtons::OK, MessageBoxIcon::Exclamation);
			return;
		}
```

```
            // Отримання зворотної матриці зворотна_A:
            Object ^ зворотна_A = XL1->Application->WorksheetFunction->MInverse(A);
            // Функція Transpose перетворює рядок вільних
            // членів у вектор:
            Object ^ вектор_B = XL1->Application->WorksheetFunction->Transpose(B);
            // Множення матриці на вектор вільних членів:
            Object ^ X =
                  XL1->Application->WorksheetFunction->MMult(зворотна_A, вектор_B);
            // Щоб відповідь набула індексованих властивостей,
            // перетворимо її в масив:
            array<Object^, 2> ^ рішення_X = (array<Object^, 2> ^)X;
            // Отримуємо двовимірний масив, індекси якого
            // починаються з одиниці:
            MessageBox::Show("Невідомі дорівнюють: x1=" + рішення_X[1, 1] + " x2=" +
                  рішення_X[2, 1] + " x3=" + рішення_X[3, 1]);
      }

      private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
            label1->Text = "x1 + 2*x2 + x3 = -1";
            label2->Text = "3*x1 - x2 - x3 = -1";
            label3->Text = "-2*x1 + 2*x2 + 3*x3 = 5";
            button1->Text = "Рішення";
            Text = "Система лінійних алгебраїчних рівнянь";
      }
private: System::Void label3_Click(System::Object^  sender, System::EventArgs^  e) {
}
};
}
```

Як видно з тексту програми, задаючи пряму матрицю, значення коефіцієнтів присвоюємо відразу при оголошенні двовимірного масиву. Аналогічно поступаємо з рядком (одновимірним масивом) вільних членів. Згідно з вимогою об'єкта `WorksheetFunction` результуючі зворотна матриця і вектор невідомих повинні бути оголошені як об'єктні змінні (типу `Object`). Спочатку обчислюємо визначник (детермінант) прямої матриці, використовуючи функцію MS Excel `MDeterm()`. Якщо пряма матриця погано обумовлена, тобто визначник дорівнює `0`, то виходимо з процедури і повідомляємо користувачеві в діалоговому вікні `MessageBox`, що система не має рішення (рис. 7.10).

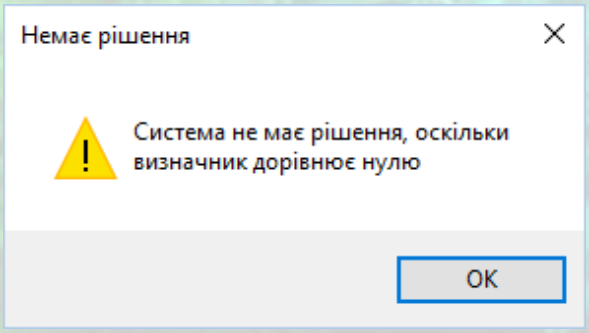

Рисунок 7.10 – Повідомлення про відсутність рішення

Якщо визначник матриці не дорівнює `0`, то з допомогою функції MS Excel `MInverse()` знаходимо обернену матрицю. Тепер рядок (одновимірний масив)

вільних членів слід перетворити у вектор, для цієї мети використовуємо функцію MS Excel Transpose() (тобто транспонувати масив B). Далі обернену матрицю за допомогою функції MS Excel MMult() множимо на вектор вільних членів. У результаті функція MMult() повертає двовимірний масив, індекси якого починаються з одиниці (але не з нуля). Наступним оператором форматуємо відповідь у діалоговому вікні MessageBox.

Результат роботи програми наведений на рис. 7.11.

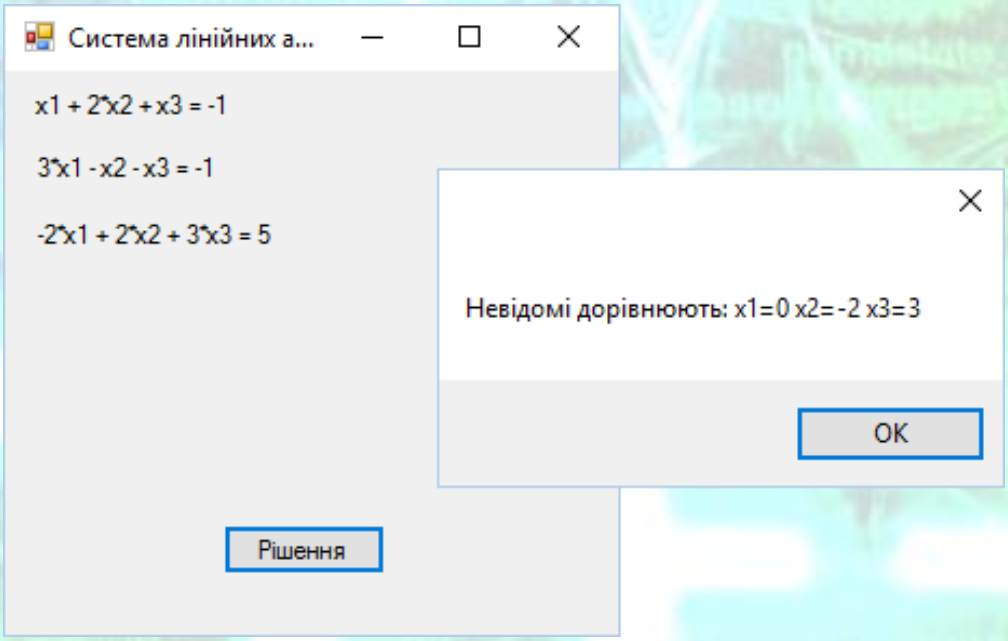

Рисунок 7.11 – Результат роботи програми з використанням математичних функцій MS Excel

Як бачимо, досить складні завдання можна вирішувати за допомогою коротенької програми завдяки зверненню до функцій MS Excel. Причому на комп'ютері, де буде працювати ця програма, зовсім не обов'язково повинен бути інстальований MS Excel. Однак інсталяційний пакет вашої програми повинен містити відповідну dll-бібліотеку функцій Microsoft Excel (файл Interop.Microsoft.Office.Interop.Excel.1.9.dll).

Повний текст даного програмного модуля наведений у лістингу 7.4.

Приклад програмної реалізації

Приклад побудови діаграм засобами MS Excel

Розглянемо можливість побудови діаграм з програми, яка написана на мові C++/CLI з використанням засобів (об'єктів компонентної бібліотеки) MS Excel.

Для вирішення цієї задачі запустимо Visual Studio і в вікні Створення проекту (New Project) виберемо в середовищі CLR вузла Visual C++ додаток

шаблону Windows Forms для C++. У проектовану екранну форму з Панелі елементів (Toolbox) перенесемо:

1) елемент Panel (об'єкт panel1), а в ньому розмістимо елемент DataGridView для відображення табличних даних;
2) елемент Panel (об'єкт panel2), а в ньому розмістимо два елементи Button (об'єкти button1 та button2);
3) елемент Panel (об'єкт panel3), а в ньому розмістимо три елементи Label, три елементи TextBox та одну кнопку Button;
4) два елементи діалогових вікон SaveFileDialog (об'єкти saveFileDialog1 та saveFileDialog2).

Після цього вікно Конструктора (Design) повинно мати наступний вигляд (рис. 7.12).

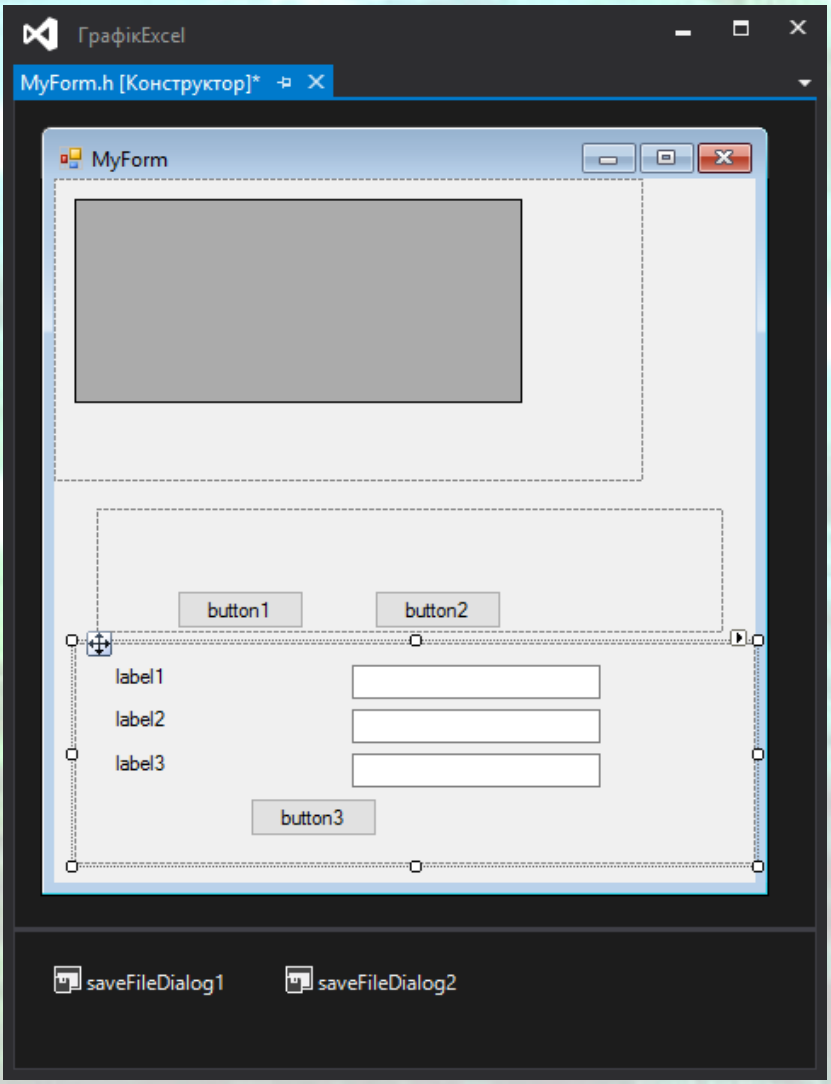

Рисунок 7.12 – Вікно конструктора форми після додавання всіх елементів

У поточний проект підключаємо бібліотеку об'єктів MS Excel. Для цього в меню Проект (Project) виберемо команду Додати посилання... (Add

Reference...), потім на вкладці COM двічі клацнемо на посиланні Microsoft Excel 16.0 Object Library (рис. 7.7).

Далі додаємо програмний код наведений нижче.

```
#pragma endregion
            DataTable^ Таблиця;  // Об'явлення объекта "Таблиця"
            DataSet^ НабірДаних; // Об'явлення объекта "НабірДаних"
            DataColumn^ Стовпчик1; // Об'явлення объекта Стовпчик1
            DataColumn^ Стовпчик2; // Об'явлення объекта Стовпчик2

      private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e) {
            // Створюємо екземпляр класу Excel::Application:
            Microsoft::Office::Interop::Excel::Application ^ XL1 = gcnew
                  Microsoft::Office::Interop::Excel::Application();
            XL1->Visible = true;
            // Задаємо параметр за замовчуванням для його подальшого
            // використання у відповідних методах:
            Object ^ t = Type::Missing;
            // Створюємо нову книгу MS Excel:
            Microsoft::Office::Interop::Excel::Workbook ^ Книга = XL1->Workbooks->Add(t);
            // Об'являємо аркуші в книзі:
            Microsoft::Office::Interop::Excel::Sheets ^ Аркуші = Книга->Worksheets;
            // Вибираємо перший аркуш:
            Microsoft::Office::Interop::Excel::Worksheet ^ Аркуш =
                  (Microsoft::Office::Interop::Excel::Worksheet ^)Аркуші->Item[1];
            // Якщо хочемо додати ще один аркуш (четвертий) до вже існуючих:
            // _Worksheet ^ Аркуш = safe_cast<_Worksheet^>(Аркуші->Item[ (Object^)1 ]);
            // Записуємо дані у відповідних комірках:
            String ^ colA;
            String ^ colB;
            // Записуємо заголовки стовпчиків на лист Excel
            Аркуш->Range["A1", t]->Value2 = Стовпчик1->Caption;
            Аркуш->Range["B1", t]->Value2 = Стовпчик2->Caption;
            // Записуємо таблицю на лист Excel
            for (int i = 0; i < Таблиця->Rows->Count; i++)
            {
                  colA = "A" + (i+2).ToString();
                  colB = "B" + (i+2).ToString();
                  Аркуш->Range[colA, t]->Value2 = dataGridView1->Rows[i]->
                        Cells[0]->Value;
                  Аркуш->Range[colB, t]->Value2 = dataGridView1->Rows[i]->
                        Cells[1]->Value;
            }
            // Замовляємо побудову діаграми (графіка) з параметрами за замовчуванням:
            Microsoft::Office::Interop::Excel::Chart ^ Графік =
            (Microsoft::Office::Interop::Excel::Chart ^)XL1->Charts->Add(t, t, t, t);
            // Задаємо діапазон значень для побудови графіку:
            Графік->SetSourceData(Аркуш->Range["A2", colB],
                  Microsoft::Office::Interop::Excel::XlRowCol::xlColumns);
            // Задаємо тип графіка "стовпчикова діаграма" (гістограма):
            Графік->ChartType =
                  Microsoft::Office::Interop::Excel::XlChartType::xlColumnClustered;
            Графік->HasLegend = false; // Відключаємо легенду графіка
            Графік->HasTitle = true; // Графік має заголовок
            Графік->ChartTitle->Caption = Таблиця->TableName;
            // Підпис осі X:
            Microsoft::Office::Interop::Excel::Axis ^ ГоризонтальнаОсь =
                  (Microsoft::Office::Interop::Excel::Axis^)Графік->Axes(
                  Microsoft::Office::Interop::Excel::XlAxisType::xlCategory,
```

```
                Microsoft::Office::Interop::Excel::XlAxisGroup::xlPrimary);
        ГоризонтальнаОсь->HasTitle = true;
        ГоризонтальнаОсь->AxisTitle->Text = Стовпчик1->Caption;
        // Підпис осі Y:
        Microsoft::Office::Interop::Excel::Axis ^ ВертикальнаОсь =
                (Microsoft::Office::Interop::Excel::Axis^)Графік->Axes(
                Microsoft::Office::Interop::Excel::XlAxisType::xlValue,
                Microsoft::Office::Interop::Excel::XlAxisGroup::xlPrimary);
        ВертикальнаОсь->HasTitle = true;
        ВертикальнаОсь->AxisTitle->Text = Стовпчик2->Caption;
        // Збереження графіка в растровому файлі:
        if (MessageBox::Show("Зберегти діаграму?", "Діаграма",
                MessageBoxButtons::YesNo, MessageBoxIcon::Question) ==
                System::Windows::Forms::DialogResult::Yes)
        {
                if (saveFileDialog2->ShowDialog() ==
                        System::Windows::Forms::DialogResult::OK)
                {
                        // Якщо користувач натиснув Зберегти, зберігаємо діаграму
                        XL1->ActiveChart->Export(saveFileDialog2->FileName, t, t);
                }
        }
}

private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
        // Задаємо написи на кнопках
        button3->Text = "Створити таблицю";
        button1->Text = "Створити діаграму Excel";
        button2->Text = "Зберегти таблицю у файл xml";
        // Задаємо заголовок вікна
        Text = "Створення діаграми в MS Excel";
        // Задаємо висоту кнопок
        button1->Height = 25;
        button2->Height = button1->Height;
        button3->Height = button1->Height;
        // Задаємо написи на мітках Lable
        label1->Text = "Назва таблиці";
        label2->Text = "Ім'я 1-го стовпчика";
        label3->Text = "Ім'я 2-го стовпчика";
        // Вирівнюємо кнопки в межах об'єктів Panel
        button1->Dock = DockStyle::Bottom;
        button2->Dock = DockStyle::Bottom;
        button3->Dock = DockStyle::Bottom;
        // Вирівнюємо об'кти Lable та TextBox в межах об'єкту panel3
        label1->Width = label3->Width;
        label2->Width = label3->Width;
        label1->Top = 10; label1->Left = 10;
        label2->Top = (label1->Height + 10) + 10; label2->Left = 10;
        label3->Top = (label1->Height + 10) * 2 + 10; label3->Left = 10;
        textBox1->Top = label1->Top; textBox1->Left = label3->Width + 30;
        textBox2->Top = label2->Top; textBox2->Left = label3->Width + 30;
        textBox3->Top = label3->Top; textBox3->Left = label3->Width + 30;
        textBox1->Width = 110;
        textBox2->Width = textBox1->Width;
        textBox3->Width = textBox1->Width;
        // Задаємо ширину форми з урахуванням ширини об'єктів Lable та TextBox
        Width = 10 + label1->Width + 20 + textBox1->Width + 60;
        // Задаємо висоту panel3 з урахуванням висоти об'єктів Lable, TextBox, button3
        panel3->Height = (label1->Height + 10) * 3 + button3->Height + 30;
        // Задаємо висоту panel2
        panel2->Height = button1->Height * 2;
```

```
            // Вирівнюємо об'єкти Panel в межах форми
            panel2->Dock = DockStyle::Bottom;
            panel3->Dock = DockStyle::Bottom;
            dataGridView1->Dock = DockStyle::Fill;
            panel1->Dock = DockStyle::Fill;
            // Властивіст DockStyle::Fill працює не завжди коректно
            //щоб виправити ситуацію можна скористатись "чарівним оператором",
            //який встановить для проблемного об'єкта індекс 0 властивості
            //SetChildIndex батьківського об'єкту
            this->Controls->SetChildIndex(panel1, 0);//////////////////////////////////
            // Приховуємо панель panel2 з кнопками побудови діаграми та збереження таблиці
            panel2->Visible = false;
            // Встановлюємо фільтр для збереження таблиці у файл XML
            saveFileDialog1->Filter = "Файли XML(*.xml)|*.xml|Всі файли (*.*)|*.*";
            // Встановлюємо фільтр для збереження діаграми у файл JPG
            saveFileDialog2->Filter = "Файли зображень JPG(*.jpg)|*.jpg|
                  Всі файли (*.*)|*.*";
      }

private: System::Void button2_Click(System::Object^  sender, System::EventArgs^  e) {
      // Зберегти файл *.xml:
      if (saveFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)
      {
            // Якщо користувач натиснув Зберегти, зберігаємо тиблицю
            НабірДаних->WriteXml(saveFileDialog1->FileName);
      }
}

private: System::Void button3_Click(System::Object^  sender, System::EventArgs^  e) {
      Таблиця = gcnew DataTable();
      НабірДаних = gcnew DataSet();
      // Зчитуємо назви стовпчиків з полів textBox
      Стовпчик1 = Таблиця->Columns->Add(textBox2->Text);
      Стовпчик2 = Таблиця->Columns->Add(textBox3->Text);
      // Зчитуємо назву таблиці з поля textBox1
      Таблиця->TableName = textBox1->Text;
      // Пов'язуємо об'єкт dataGridView1 з Таблицею
      dataGridView1->DataSource = Таблиця;
      // Додати об'єкт Таблиця в DataSet
      НабірДаних->Tables->Add(Таблиця);
      // Відображаємо панель з кнопками
      panel2->Visible = true;
}
};
}
```

Код даного програмного модуля досить об’ємний, що пов’язане з програмуванням вирівнювання елементів інтерфейсу, та певною універсальністю даної програми. Програма пропоноє користувачу задати ім’я таблиці та імена двох стовпчиків даних. Після цього за допомогою натискання кнопки створюється таблиця даних з двома стовпцями, в які можна вводити довільні дані з будь-якою кількістю рядків. На основі введених двох стовпчиків даних і буде виконуватись побудова діаграмив MS Excel.

У обробнику події MyForm_Load присвоюються значення різним властивостям об’єктів форми. Більшість цих властивостей зручніше задавати в режимі проектування у вікні Властивості (Properties) для відповідного

об'єкту. Також у обробнику події `MyForm_Load` деякі властивості задаються у вигляді залежностей для досягнення ідеального розташування та розмірів елементів інтерфейсу. Слід звернути увагу, що при встановленні властивості `Dock` у значення `Fill` (заповнення) для об'єктів, ця властивість часто працює некоректно і займає площу і інших вже вирівняних об'єктів. Щоб об'єкти інтерфейсу після вирівнювання не перекривали один одного, потрібно додати наступний рядок коду:

```
this->Controls->SetChildIndex(panel1, 0);
```

де `this` – це батьківський об'єкт-контейнер в межах якого вирівнюються інші об'єкти (в даному випадку це форма); `SetChildIndex()` – функція, в яку в якості першого параметру передаємо ім'я об'єкту, який має властивість `Dock` зі значенням `Fill` (заповнення), в нашому випадку це `panel1`.

Після запуску програми ми побачимо вікно форми, в якому ще немає таблиці. Відповідно немає даних, за якими потрібно будувати діаграму, або зберігати таблицю у файл XML. За ці дії в нас відповідають дві кнопки, які розташовані на панелі `panel2`. Тому при запуску форми ми приховали цю панель з двома кнопками (рис. 7.13).

```
// Приховуємо панель panel2 з кнопками побудови діаграми та збереження таблиці
panel2->Visible = false;
```

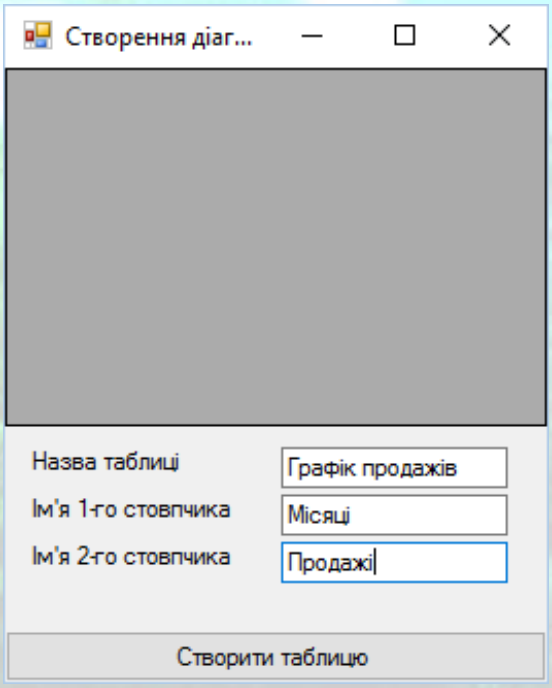

Рисунок 7.13 – Вікно програми після запуску з введеними назвами стовпчиків та таблиці

Коли користувач введе імена стовпчиків та назву таблиці і натисне кнопку `Створити таблицю` (обробник події `button3_Click`), у вікні з'явиться пуста

таблиця, а також панель з двома кнопками. В таблицю потрібно ввести дані за якими і будуватимемо діаграму (рис. 7.14).

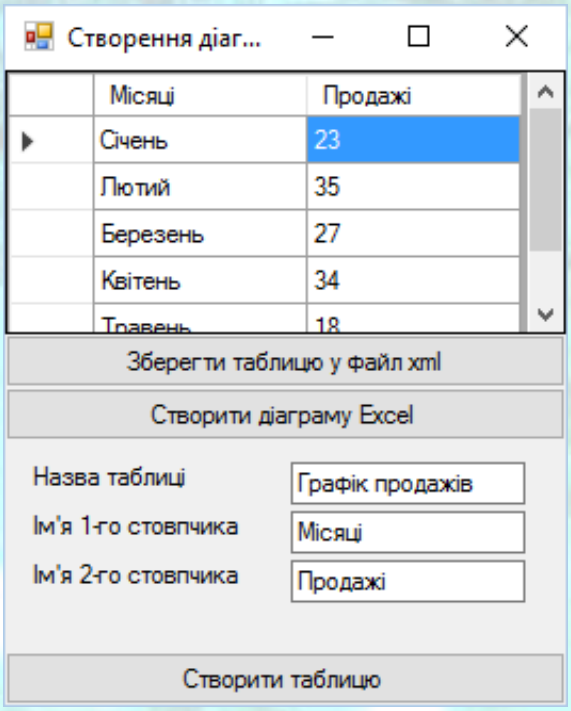

Рисунок 7.14 – Вікно програми після введення таблиці даних

При натисканні кнопки `Зберегти таблицю у файл XML` вікликається обробник події `button2_Click`, в якому передбачене виведення стандартного діалогового вікна збереження файлу для можливості обрання місця збереження та імені файлу користувачем (рис. 7.15).

```
// Зберегти файл *.xml:
if (saveFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)
{
        // Якщо користувач натиснув Зберегти, зберігаємо тиблицю
        НабірДаних->WriteXml(saveFileDialog1->FileName);
}
```

За діалогове вікно збереження файлу відповідає об’єкт `saveFileDialog1`, в якому встановлене значення фільтру для відображення за замовчуванням лише файлів XML.

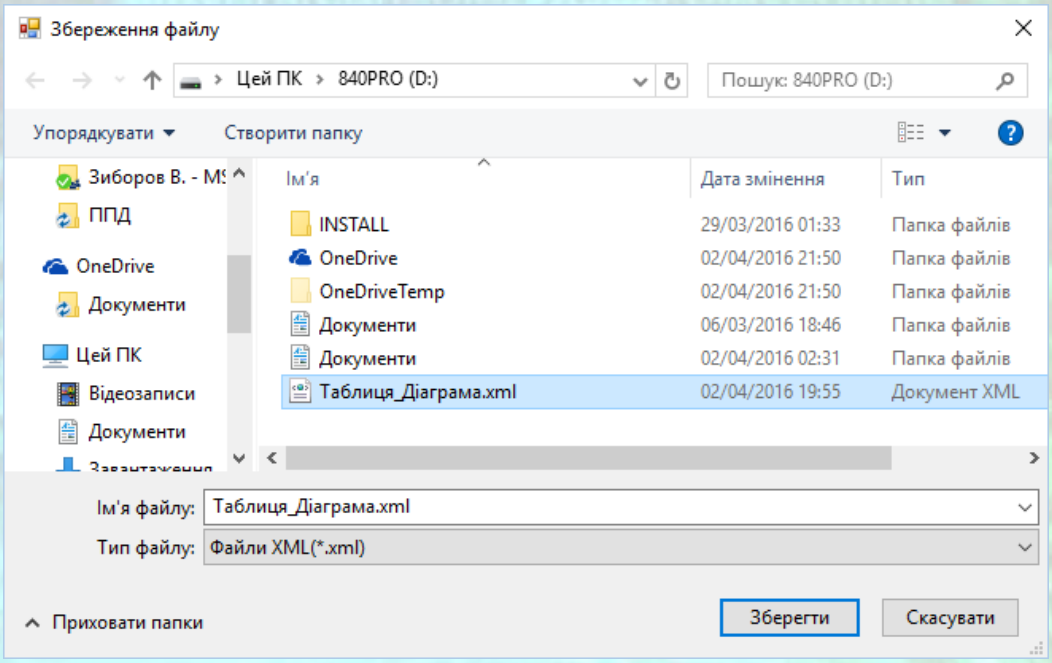

Рисунок 7.15 – Діалогове вікно збереження таблиці у файл XML

Після збереження таблиці даних, їх можна перегрянути у сторонній програмі, наприклад, у інтернет браузері.

Кнопка Створити діаграму Excel безпосередньо відповідає за створення діаграми (обробник події button1_Click). Для можливості створення діаграми ми спочатку записуємо нашу таблицю на аркуш Excel. За це відповідає наступна частина коду.

```
// Створюємо екземпляр класу Excel::Application:
Microsoft::Office::Interop::Excel::Application ^ XL1 = gcnew
       Microsoft::Office::Interop::Excel::Application();
// Відображення вікна Excel
XL1->Visible = true;
// Задаємо параметр за замовчуванням для його подальшого
// використання у відповідних методах:
Object ^ t = Type::Missing;
// Створюємо нову книгу MS Excel:
Microsoft::Office::Interop::Excel::Workbook ^ Книга = XL1->Workbooks->Add(t);
// Об'являємо аркуші в книзі:
Microsoft::Office::Interop::Excel::Sheets ^ Аркуші = Книга->Worksheets;
// Вибираємо перший аркуш:
Microsoft::Office::Interop::Excel::Worksheet ^ Аркуш =
       (Microsoft::Office::Interop::Excel::Worksheet ^)Аркуші->Item[1];
// Якщо хочемо додати ще один аркуш до вже існуючих:
// _Worksheet ^ Аркуш = safe_cast<_Worksheet^>(Аркуші->Item[ (Object^)1 ]);
// Записуємо дані у відповідних комірках:
String ^ colA;
String ^ colB;
// Записуємо заголовки стовпчиків на лист Excel
Аркуш->Range["A1", t]->Value2 = Стовпчик1->Caption;
Аркуш->Range["B1", t]->Value2 = Стовпчик2->Caption;
// Записуємо таблицю на лист Excel
for (int i = 0; i < Таблиця->Rows->Count; i++)
{
```

```
        colA = "A" + (i+2).ToString();
        colB = "B" + (i+2).ToString();
        Аркуш->Range[colA, t]->Value2 = dataGridView1->Rows[i]->Cells[0]->Value;
        Аркуш->Range[colB, t]->Value2 = dataGridView1->Rows[i]->Cells[1]->Value;
}
```

При цьому ми робимо видимим вікно MS Excel і безпосередньо спостерігаємо заповнення таблиці на аркуші книги Excel. Після створення таблиці даних починається безпосередньо побудова діаграми.

```
// Замовляємо побудову діаграми (графіка) з параметрами за замовчуванням:
Microsoft::Office::Interop::Excel::Chart ^ Графік =
        (Microsoft::Office::Interop::Excel::Chart ^)XL1->Charts->Add(t, t, t, t);
// Задаємо діапазон значень для побудови графіку:
Графік->SetSourceData(Аркуш->Range["A2", colB],
        Microsoft::Office::Interop::Excel::XlRowCol::xlColumns);
// Задаємо тип графіка "стовпчикова діаграма" (гістограма):
Графік->ChartType = Microsoft::Office::Interop::Excel::XlChartType::xlColumnClustered;
// Відключаємо легенду графіка:
Графік->HasLegend = false;
// Графік має заголовок:
Графік->HasTitle = true;
Графік->ChartTitle->Caption = Таблиця->TableName;
// Підпис осі X:
Microsoft::Office::Interop::Excel::Axis ^ ГоризонтальнаОсь =
        (Microsoft::Office::Interop::Excel::Axis^)Графік->Axes(
        Microsoft::Office::Interop::Excel::XlAxisType::xlCategory,
        Microsoft::Office::Interop::Excel::XlAxisGroup::xlPrimary);
ГоризонтальнаОсь->HasTitle = true;
ГоризонтальнаОсь->AxisTitle->Text = Стовпчик1->Caption;
// Підпис осі Y:
Microsoft::Office::Interop::Excel::Axis ^ ВертикальнаОсь =
        (Microsoft::Office::Interop::Excel::Axis^)Графік->Axes(
        Microsoft::Office::Interop::Excel::XlAxisType::xlValue,
        Microsoft::Office::Interop::Excel::XlAxisGroup::xlPrimary);
ВертикальнаОсь->HasTitle = true;
ВертикальнаОсь->AxisTitle->Text = Стовпчик2->Caption;
```

Результатом буде поява стовпчикової діаграми у вікні MS Excel з заголовком та підписами вісей (рис. 7.16).

Зрозуміло, що тип діаграми можна задавати і інший у програмному коді. При цьому різновидів діаграм набагато більше, ніж у об'єкті Chart мови C++/CLI.

У програмі також передбачена можливість збереження отриманої у MS Excel діаграми у вигляді графічного файлу з використанням діалогового вікна saveFileDialog2. За це від повідає наступний код.

```
// Збереження графіка в растровому файлі:
if (MessageBox::Show("Зберегти діаграму?", "Діаграма",
        MessageBoxButtons::YesNo, MessageBoxIcon::Question) ==
        System::Windows::Forms::DialogResult::Yes)
{
        if (saveFileDialog2->ShowDialog() == System::Windows::Forms::DialogResult::OK)
        {
```

```
            // Якщо користувач натиснув Зберегти, зберігаємо діаграму
            XL1->ActiveChart->Export(saveFileDialog2->FileName, t, t);
        }
}
```

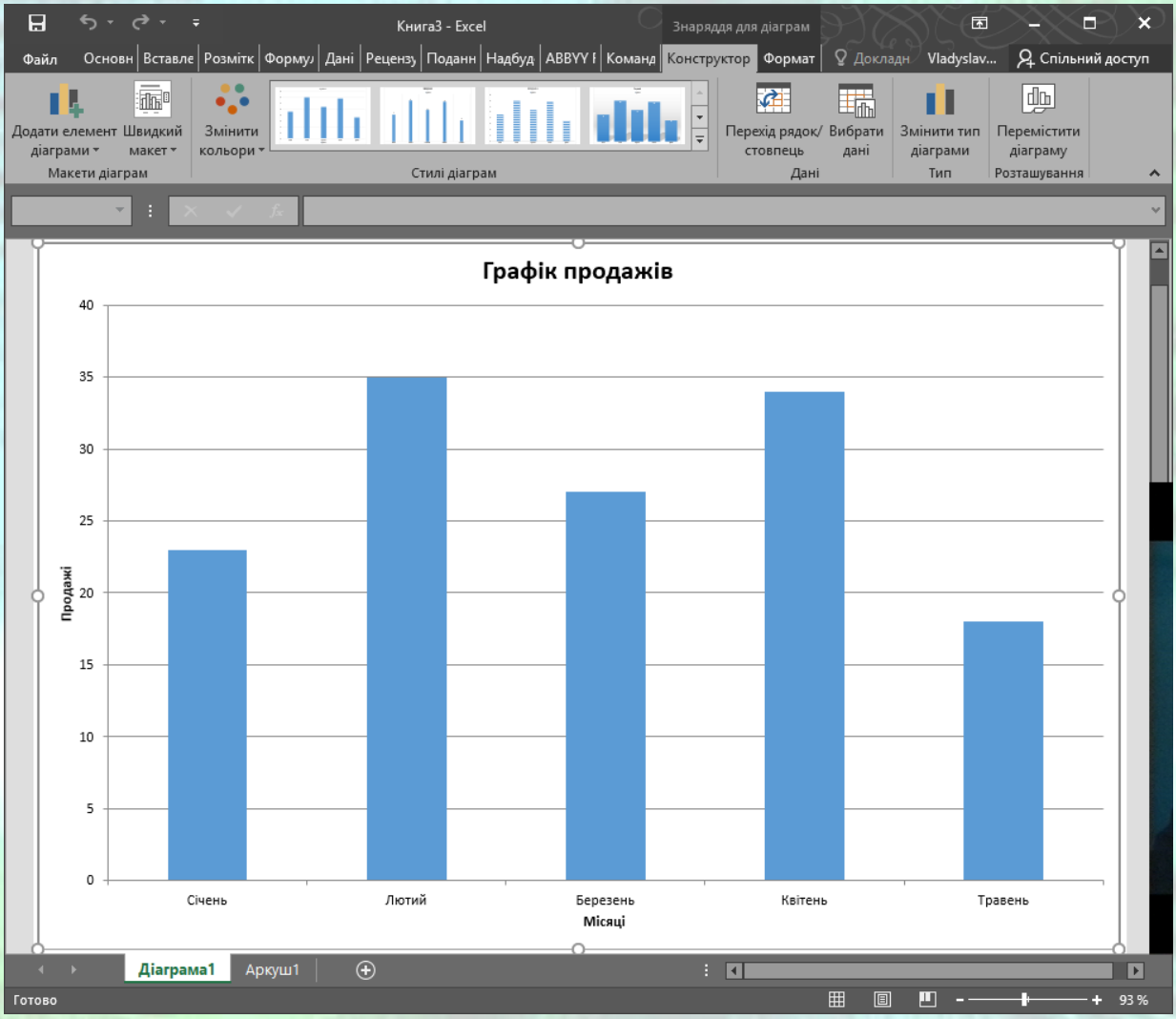

Рисунок 7.16 – Побудована діаграма у вікні MS Excel

Після побудови діаграми на екрані з'явиться діалогове вікно з запитом на збереження діаграми, створене за допомогою функції MessageBox (рис. 7.17).

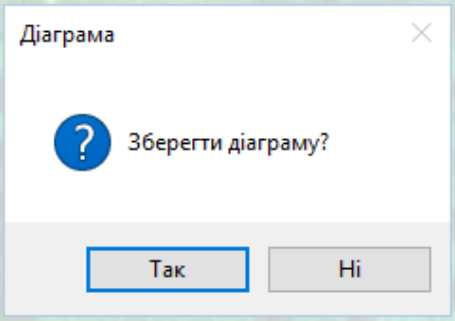

Рисунок 7.17 – Запит на збереження діаграми

Якщо користувач натисне Так, то з'явиться стандартне діалогове вікно збереження файлу (об'єкт saveFileDialog2), в якому потрібно вказати ім'я та розташування файлу JPG, в якому буде збережено створену діаграму.

В даному прикладі ми зробили вікно MS Excel видимим і спостерігали за виконанням операцій. Після цього книгу MS Excel можна зберегти, а також змінювати параметри діаграми вже безпосередньо у вікні MS Excel. Проте, якщо додаткові операції в MS Excel нам не потрібні, ми могли б, як і у попередніх прикладах не показувати вікно MS Excel, а створити в ньому діаграму, експортувати її у растрове зображення, яке б відкрили у об'єкті PictureBox форми. Тоді б все виглядало так, ніби всі дії виконані безпосередньо в нашій програмі. При цьому наша програма отримала потужний інструментарій створення діаграм MS Excel.

Повний текст даного програмного модуля наведений у лістингу 7.5.

Програмний код

Лістинг 7.1

```
// Програма дозволяє користувачеві ввести будь-які слова, речення
// в текстове поле і після натиснення відповідної кнопки перевірити
// орфографію введеного тексту. Для безпосередньої перевірки орфографії
// скористаємося функцією CheckSpelling об'єктної бібліотеки MS Word

#pragma once

namespace WindowsForms {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Сводка для MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
		}

	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::Button^  button1;
	protected:
	private: System::Windows::Forms::Panel^  panel1;
	private: System::Windows::Forms::RichTextBox^  richTextBox1;

	protected:

	private:
		/// <summary>
		/// Обязательная переменная конструктора.
		/// </summary>
		System::ComponentModel::Container ^components;
```

```
#pragma region Windows Form Designer generated code
		/// <summary>
		/// Требуемый метод для поддержки конструктора — не изменяйте
		/// содержимое этого метода с помощью редактора кода.
		/// </summary>
		void InitializeComponent(void)
		{
			this->button1 = (gcnew System::Windows::Forms::Button());
			this->panel1 = (gcnew System::Windows::Forms::Panel());
			this->richTextBox1 = (gcnew System::Windows::Forms::RichTextBox());
			this->panel1->SuspendLayout();
			this->SuspendLayout();
			//
			// button1
			//
			this->button1->Dock = System::Windows::Forms::DockStyle::Bottom;
			this->button1->Location = System::Drawing::Point(0, 238);
			this->button1->Name = L"button1";
			this->button1->Size = System::Drawing::Size(284, 23);
			this->button1->TabIndex = 1;
			this->button1->Text = L"button1";
			this->button1->UseVisualStyleBackColor = true;
			this->button1->Click += gcnew System::EventHandler(this,
&MyForm::button1_Click_1);
			//
			// panel1
			//
			this->panel1->Controls->Add(this->richTextBox1);
			this->panel1->Dock = System::Windows::Forms::DockStyle::Fill;
			this->panel1->Location = System::Drawing::Point(0, 0);
			this->panel1->Name = L"panel1";
			this->panel1->Size = System::Drawing::Size(284, 238);
			this->panel1->TabIndex = 2;
			//
			// richTextBox1
			//
			this->richTextBox1->Dock = System::Windows::Forms::DockStyle::Fill;
			this->richTextBox1->Location = System::Drawing::Point(0, 0);
			this->richTextBox1->Name = L"richTextBox1";
			this->richTextBox1->Size = System::Drawing::Size(284, 238);
			this->richTextBox1->TabIndex = 1;
			this->richTextBox1->Text = L"";
			//
			// MyForm
			//
			this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
			this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
			this->ClientSize = System::Drawing::Size(284, 261);
			this->Controls->Add(this->panel1);
			this->Controls->Add(this->button1);
			this->Name = L"MyForm";
			this->Text = L"MyForm";
			this->Load += gcnew System::EventHandler(this, &MyForm::MyForm_Load);
			this->panel1->ResumeLayout(false);
			this->ResumeLayout(false);

		}
#pragma endregion

private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
	// У пункті меню Project виберемо команду Add Reference.
```

```
        // Потім, якщо на вашому комп'ютері встановлений MS Office 2007,
        // То на вкладці COM двічі клацнемо по посиланню
        // На бібліотеку Microsoft Word 12.0 Object Libary.
        richTextBox1->Clear();
        button1->Text = "Перевірка орфографії";
        richTextBox1->TabIndex = 0;
        button1->TabIndex = 1;
        button1->Dock = System::Windows::Forms::DockStyle::Bottom;
        panel1->Dock = System::Windows::Forms::DockStyle::Fill;
        richTextBox1->Dock = System::Windows::Forms::DockStyle::Fill;
        Text = "Орфографія";
}

private: System::Void button1_Click_1(System::Object^  sender, System::EventArgs^  e) {
        auto Ворд1 = gcnew Microsoft::Office::Interop::Word::Application();
        // Ворд1-> Visible = false;
        // Змінна з "порожнім" значенням:
        System::Object  ^ t = Type::Missing;
        // Відкриваємо новий порожній документ MS Word:
        Ворд1->Documents->Add(t, t, t, t);
        // Копіюємо вміст текстового вікна в документ
        Ворд1->Selection->default = richTextBox1->Text;
        // Для VB і C # було б: Ворд1->Selection->Text
        // Безпосередня перевірка орфографії:
        Ворд1->ActiveDocument->CheckSpelling(t, t, t, t, t, t, t, t, t, t, t, t, t);
        // Копіюємо результат назад в текстове поле
        richTextBox1->Text = Ворд1->Selection->default;
        Object ^ tt = false;
        Ворд1->Documents->Close(tt, t, t);
        // Закрити документ Word без збереження:
        Ворд1->Quit(tt, t, t);
        Ворд1 = nullptr;
}
};
}
```

Лістинг 7.2

```
// Програма виведення таблиці засобами MS Word: запускається
// програма, користувач спостерігає, як запускається редактор
// MS Word і автоматично відбувається побудова таблиці

#pragma once

namespace WindowsForms {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Сводка для MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
		}

	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::Button^  button1;
	protected:

	private:
		/// <summary>
		/// Обязательная переменная конструктора.
		/// </summary>
		System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
		/// <summary>
		/// Требуемый метод для поддержки конструктора — не изменяйте
		/// содержимое этого метода с помощью редактора кода.
		/// </summary>
		void InitializeComponent(void)
		{
			this->button1 = (gcnew System::Windows::Forms::Button());
			this->SuspendLayout();
```

```
                    //
                    // button1
                    //
                    this->button1->Location = System::Drawing::Point(100, 168);
                    this->button1->Name = L"button1";
                    this->button1->Size = System::Drawing::Size(75, 23);
                    this->button1->TabIndex = 0;
                    this->button1->Text = L"button1";
                    this->button1->UseVisualStyleBackColor = true;
                    this->button1->Click += gcnew System::EventHandler(this,
&MyForm::button1_Click);
                    //
                    // MyForm
                    //
                    this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                    this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                    this->ClientSize = System::Drawing::Size(284, 261);
                    this->Controls->Add(this->button1);
                    this->Name = L"MyForm";
                    this->Text = L"MyForm";
                    this->Load += gcnew System::EventHandler(this, &MyForm::MyForm_Load);
                    this->ResumeLayout(false);

                }
#pragma endregion

        private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
                // У меню Project вкажемо команду Add Reference і на
                // Вкладці COM двічі клацнемо по посиланню на
                // Бібліотеку Microsoft Word 16.0 Object Library
                button1->Text = "Пуск"; this->Text = "Побудова таблиці";
        }

        private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e) {
                // Ініціалізуємо два рядкових масиви:
                array <String ^> ^ Імена = { "Андрій (роб)", "Світла (моб)", "Микола (дом)",
"Кафедра (роб)", "Олександр Степанович", "Сергій (дом)", "Тетяна Петрівна", "Таксі",
"Кінотеатр" };
                array <String ^> ^ Телефон = { "274-88-17", "+38 (067) 7030356", "22-345-72",
"204-82-12", "223-67-67 доп 32-67", "570-38-76", "201-72-23", "555", "216-40-22" };
                // Створюємо новий екземпляр класу Word::Application:
                auto Ворд1 = gcnew Microsoft::Office::Interop::Word::Application();
                Ворд1->Visible = true;
                // Змінна з "порожнім" значенням:
                System::Object ^ t = Type::Missing;
                // Відкриваємо новий документ MS Word:
                auto Документ = Ворд1->Documents->Add(t, t, t, t);
                // Вводимо текст в документ MS WORD з поточної позиції:
                Ворд1->Selection->TypeText("ТАБЛИЦЯ ТЕЛЕФОНІВ");
                // Параметр, який вказує чи показувати межі комірок:
                System::Object ^ t1 = Microsoft::Office::Interop::
                        Word::WdDefaultTableBehavior::wdWord9TableBehavior;
                // Параметр, який вказує чи буде додаток Word автоматично
                // змінювати розмір комірок у таблиці для підгонки вмісту:
                System::Object ^ t2 =
Microsoft::Office::Interop::Word::WdAutoFitBehavior::wdAutoFitContent;
                // Створюємо таблицю з 9 рядків і 2 стовпців:
                Ворд1->ActiveDocument->Tables->Add(Ворд1->Selection->Range, 9, 2, t1, t2);
                // Заповнювати комірки таблиці можна так:
                for (int i = 1; i <= 9; i++)
                {
```

```
                    Ворд1->ActiveDocument->Tables[1]->Cell(i, 1)->default-
>InsertAfter(Імена[i - 1]);
                    Ворд1->ActiveDocument->Tables[1]->Cell(i, 2)->default-
>InsertAfter(Телефон[i - 1]);
                    // Програмуючи на C# ми написали б:
                    // Ворд1.ActiveDocument.Tables[1].Cell(i, 2).Range.InsertAfter(Tel[i -
1]);
                }
                // Призначаємо одиниці вимірювання в документі додатка MS Word:
                Object ^ t3 = Microsoft::Office::Interop::Word::WdUnits::wdLine;
                // Параметр, який вказує на дев'ятий рядок у документі MS Word:
                Object ^ рядок9 = 9;
                // Перевести поточну позицію (Selection) за межі таблиці,
                // (В дев'ятий рядок), щоб тут вивести будь-який текст:
                Ворд1->Selection->MoveDown(t3, рядок9, t);
                // І тут друкуємо наступний текст:
                Ворд1->Selection->TypeText("Який-небудь текст після таблиці");
                // Зберігати документ немає сенсу, але це вирішить користувач:
                // або можна задати безпосередньо шлях та ім'я файлу
                Object ^ ім’яФайла = "D:\\Таблиця.docx";
                // і одразу зберегти
                Ворд1->ActiveDocument->SaveAs(ім’яФайла, t, t, t, t, t, t, t, t, t, t, t, t,
t, t, t);
            }
        };
}
```

Лістинг 7.3

```
// Програма використовує фінансову функцію Pmt() об'єктної бібліотеки
// MS Excel для обчислення суми періодичного платежу на основі
// сталості сум платежів і постійності процентної ставки

#pragma once
// ~ ~ ~ ~ ~ ~ ~ ~ ~
// Для підключення бібліотеки об'єктів MS Excel в пункті меню Project
// Виберемо команду Add Reference. Потім, якщо на вашому комп'ютері
// Встановлений MS Office 2016, то на вкладці COM двічі клацнемо по
// Посиланням на бібліотеку Microsoft Excel 16.0 Object Library

namespace WindowsForms {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Сводка для MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
		}

	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::Label^  label1;
	private: System::Windows::Forms::Label^  label2;
	private: System::Windows::Forms::Label^  label3;
	private: System::Windows::Forms::TextBox^  textBox3;
	private: System::Windows::Forms::TextBox^  textBox1;
	private: System::Windows::Forms::TextBox^  textBox2;
	private: System::Windows::Forms::Button^  button1;
	protected:

	private:
		/// <summary>
		/// Обязательная переменная конструктора.
		/// </summary>
```

```
        System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
        /// <summary>
        /// Требуемый метод для поддержки конструктора — не изменяйте
        /// содержимое этого метода с помощью редактора кода.
        /// </summary>
        void InitializeComponent(void)
        {
            this->label1 = (gcnew System::Windows::Forms::Label());
            this->label2 = (gcnew System::Windows::Forms::Label());
            this->label3 = (gcnew System::Windows::Forms::Label());
            this->textBox3 = (gcnew System::Windows::Forms::TextBox());
            this->textBox1 = (gcnew System::Windows::Forms::TextBox());
            this->textBox2 = (gcnew System::Windows::Forms::TextBox());
            this->button1 = (gcnew System::Windows::Forms::Button());
            this->SuspendLayout();
            //
            // label1
            //
            this->label1->AutoSize = true;
            this->label1->Location = System::Drawing::Point(12, 20);
            this->label1->Name = L"label1";
            this->label1->Size = System::Drawing::Size(35, 13);
            this->label1->TabIndex = 0;
            this->label1->Text = L"label1";
            //
            // label2
            //
            this->label2->AutoSize = true;
            this->label2->Location = System::Drawing::Point(12, 51);
            this->label2->Name = L"label2";
            this->label2->Size = System::Drawing::Size(35, 13);
            this->label2->TabIndex = 1;
            this->label2->Text = L"label2";
            //
            // label3
            //
            this->label3->AutoSize = true;
            this->label3->Location = System::Drawing::Point(12, 88);
            this->label3->Name = L"label3";
            this->label3->Size = System::Drawing::Size(35, 13);
            this->label3->TabIndex = 2;
            this->label3->Text = L"label3";
            //
            // textBox3
            //
            this->textBox3->Location = System::Drawing::Point(141, 88);
            this->textBox3->Name = L"textBox3";
            this->textBox3->Size = System::Drawing::Size(100, 20);
            this->textBox3->TabIndex = 5;
            //
            // textBox1
            //
            this->textBox1->Location = System::Drawing::Point(141, 20);
            this->textBox1->Name = L"textBox1";
            this->textBox1->Size = System::Drawing::Size(100, 20);
            this->textBox1->TabIndex = 6;
            //
            // textBox2
            //
```

```
                    this->textBox2->Location = System::Drawing::Point(141, 51);
                    this->textBox2->Name = L"textBox2";
                    this->textBox2->Size = System::Drawing::Size(100, 20);
                    this->textBox2->TabIndex = 7;
                    //
                    // button1
                    //
                    this->button1->Location = System::Drawing::Point(99, 181);
                    this->button1->Name = L"button1";
                    this->button1->Size = System::Drawing::Size(75, 23);
                    this->button1->TabIndex = 8;
                    this->button1->Text = L"button1";
                    this->button1->UseVisualStyleBackColor = true;
                    this->button1->Click += gcnew System::EventHandler(this,
&MyForm::button1_Click);
                    //
                    // MyForm
                    //
                    this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                    this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                    this->ClientSize = System::Drawing::Size(284, 261);
                    this->Controls->Add(this->button1);
                    this->Controls->Add(this->textBox2);
                    this->Controls->Add(this->textBox1);
                    this->Controls->Add(this->textBox3);
                    this->Controls->Add(this->label3);
                    this->Controls->Add(this->label2);
                    this->Controls->Add(this->label1);
                    this->Name = L"MyForm";
                    this->Text = L"MyForm";
                    this->Load += gcnew System::EventHandler(this, &MyForm::MyForm_Load);
                    this->ResumeLayout(false);
                    this->PerformLayout();

                }
#pragma endregion

private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
                this->Text = "Розрахунок щомісячних платежів";
                label1->Text = "Річна ставка у %";
                label2->Text = "Термін в місяцях";
                label3->Text = "Розмір кредиту";
                textBox1->Clear();
                textBox2->Clear();
                textBox3->Clear();
                button1->Text = "Розрахунок";
        }

private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e) {
        try
        {
                auto XL = gcnew Microsoft::Office::Interop::Excel::Application();
                // Змінна з "порожнім" значенням:
                auto t = Type::Missing;
                // Отримуємо розмір місячного платежу:
                double pay = XL->WorksheetFunction->Pmt(
                        (Convert::ToDouble(textBox1->Text))/1200,
                        Convert::ToDouble(textBox2->Text),
                        Convert::ToDouble(textBox3->Text), t, t);
                // АБО, якщо використовувати функцію Pmt()
                // з Microsoft VisualBasic:
```

```
                // Double FV = 0;
                // Microsoft::VisualBasic::DueDate dt =
                // Microsoft::VisualBasic::DueDate::EndOfPeriod;
                // Double pay = Microsoft::VisualBasic::Financial::Pmt(
                // (Convert::ToDouble(textBox1->Text))/1200,
                // Convert::ToDouble(textBox2->Text),
                // Convert::ToDouble(textBox3->Text), FV, dt);
                auto Рядок = String::Format(
                        "Кожен місяць слід платити {0:$ #.##} доларів",
                        Math::Abs(pay));
                MessageBox::Show(Рядок);
                XL->Quit();
        }
        catch (Exception ^ Ситуація)
        {
                MessageBox::Show(Ситуація->Message, "Помилка",
                        MessageBoxButtons::OK, MessageBoxIcon::Exclamation);
        }
}
};
}
```

Лістинг 7.4

```
// Програма вирішує систему рівнянь за допомогою функцій об'єктної бібліотеки MS Excel

#pragma once
// Для підключення бібліотеки об'єктів MS Excel в пункті меню Project
// виберемо команду Add Reference. Потім, якщо на вашому комп'ютері
// встановлений MS Office 2016, то на вкладці COM двічі клацнемо по
// посиланню на бібліотеку Microsoft Excel 16.0 Object Library

namespace WindowsForms {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Сводка для MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
		}

	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::Button^  button1;
	private: System::Windows::Forms::Label^  label1;
	private: System::Windows::Forms::Label^  label2;
	private: System::Windows::Forms::Label^  label3;
	protected:

	private:
		/// <summary>
		/// Обязательная переменная конструктора.
		/// </summary>
		System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
		/// <summary>
		/// Требуемый метод для поддержки конструктора — не изменяйте
```

```
		/// содержимое этого метода с помощью редактора кода.
		/// </summary>
		void InitializeComponent(void)
		{
			this->button1 = (gcnew System::Windows::Forms::Button());
			this->label1 = (gcnew System::Windows::Forms::Label());
			this->label2 = (gcnew System::Windows::Forms::Label());
			this->label3 = (gcnew System::Windows::Forms::Label());
			this->SuspendLayout();
			//
			// button1
			//
			this->button1->Location = System::Drawing::Point(101, 210);
			this->button1->Name = L"button1";
			this->button1->Size = System::Drawing::Size(75, 23);
			this->button1->TabIndex = 0;
			this->button1->Text = L"button1";
			this->button1->UseVisualStyleBackColor = true;
			this->button1->Click += gcnew System::EventHandler(this,
&MyForm::button1_Click);
			//
			// label1
			//
			this->label1->AutoSize = true;
			this->label1->Location = System::Drawing::Point(12, 9);
			this->label1->Name = L"label1";
			this->label1->Size = System::Drawing::Size(35, 13);
			this->label1->TabIndex = 1;
			this->label1->Text = L"label1";
			//
			// label2
			//
			this->label2->AutoSize = true;
			this->label2->Location = System::Drawing::Point(12, 38);
			this->label2->Name = L"label2";
			this->label2->Size = System::Drawing::Size(35, 13);
			this->label2->TabIndex = 2;
			this->label2->Text = L"label2";
			//
			// label3
			//
			this->label3->AutoSize = true;
			this->label3->Location = System::Drawing::Point(12, 69);
			this->label3->Name = L"label3";
			this->label3->Size = System::Drawing::Size(35, 13);
			this->label3->TabIndex = 3;
			this->label3->Text = L"label3";
			this->label3->Click += gcnew System::EventHandler(this,
&MyForm::label3_Click);
			//
			// MyForm
			//
			this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
			this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
			this->ClientSize = System::Drawing::Size(284, 261);
			this->Controls->Add(this->label3);
			this->Controls->Add(this->label2);
			this->Controls->Add(this->label1);
			this->Controls->Add(this->button1);
			this->Name = L"MyForm";
			this->Text = L"MyForm";
```

```
                this->Load += gcnew System::EventHandler(this, &MyForm::MyForm_Load);
                this->ResumeLayout(false);
                this->PerformLayout();
            }
#pragma endregion

        private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e) {
            // Вирішуємо систему
            // x1 + 2*x2 + x3 = -1
            // 3*x1 - x2 - x3 = -1
            // -2*x1 + 2*x2 + 3*x3 = 5
            // Для цієї системи пряма матриця буде мати вигляд:
            // array<double,2> ^ A = gcnew array<double,2>(n, n);
            array<double, 2> ^ A = { { 1, 2, 1 },
                                     { 3, -1, -1 },
                                     { -2, 2, 3 } };
            // Рядок вільних членів:
            // array<double> ^ B = gcnew array<double>(n);
            // Вільні члени:
            array<double> ^ B = { -1, -1, 5 };
            // Створення екземпляра класу Excel::Application:
            auto XL1 = gcnew Microsoft::Office::Interop::Excel::Application();
            // Обчислення детермінанта матриці A
            double визначник_A = XL1->Application->WorksheetFunction->MDeterm(A);
            // Якщо визначник_A != 0, то вихід з процедури:
            if (визначник_A == 0)
            {
                MessageBox::Show("Система не має рішення, оскільки\n" +
                    "визначник дорівнює нулю", "Немає рішення",
MessageBoxButtons::OK, MessageBoxIcon::Exclamation);
                return;
            }
            // Отримання зворотної матриці зворотна_A:
            Object ^ зворотна_A = XL1->Application->WorksheetFunction->MInverse(A);
            // Функція Transpose перетворює рядок вільних
            // членів у вектор:
            Object ^ вектор_B = XL1->Application->WorksheetFunction->Transpose(B);
            // Множення матриці на вектор вільних членів:
            Object ^ X =XL1->Application->WorksheetFunction->MMult(зворотна_A, вектор_B);
            // Щоб відповідь набула індексованих властивостей,
            // перетворимо її в масив:
            array<Object^, 2> ^ рішення_X = (array<Object^, 2> ^)X;
            // Отримуємо двовимірний масив, індекси якого
            // починаються з одиниці:
            MessageBox::Show("Невідомі дорівнюють: x1=" + рішення_X[1, 1] +
                " x2=" + рішення_X[2, 1] + " x3=" + рішення_X[3, 1]);
        }

        private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
            label1->Text = "x1 + 2*x2 + x3 = -1";
            label2->Text = "3*x1 - x2 - x3 = -1";
            label3->Text = "-2*x1 + 2*x2 + 3*x3 = 5";
            button1->Text = "Рішення";
            Text = "Система лінійних алгебраїчних рівнянь";
        }
};
}
```

Лістинг 7.5

```
// Програма побудови діаграми на основі двох стовпчиків даних,
//які ввів користувач, засобами MS Excel з можливістю збереження
//отриманої діаграми у растровому зображенні

#pragma once
namespace WindowsForms {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;
	//using namespace Microsoft::Office::Interop::Excel;

	/// <summary>
	/// Сводка для MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
		}

	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::Panel^  panel1;
	protected:

	private: System::Windows::Forms::Panel^  panel3;
	private: System::Windows::Forms::Button^  button3;
	private: System::Windows::Forms::TextBox^  textBox3;
	private: System::Windows::Forms::TextBox^  textBox2;
	private: System::Windows::Forms::TextBox^  textBox1;
	private: System::Windows::Forms::Label^  label3;
	private: System::Windows::Forms::Label^  label2;
	private: System::Windows::Forms::Label^  label1;
	private: System::Windows::Forms::DataGridView^  dataGridView1;
	private: System::Windows::Forms::Panel^  panel2;
	private: System::Windows::Forms::Button^  button2;
	private: System::Windows::Forms::Button^  button1;
	private: System::Windows::Forms::SaveFileDialog^  saveFileDialog1;
	private: System::Windows::Forms::SaveFileDialog^  saveFileDialog2;
	private:
```

```
        /// <summary>
        /// Обязательная переменная конструктора.
        /// </summary>
        System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
        /// <summary>
        /// Требуемый метод для поддержки конструктора — не изменяйте
        /// содержимое этого метода с помощью редактора кода.
        /// </summary>
        void InitializeComponent(void)
        {
            this->panel1 = (gcnew System::Windows::Forms::Panel());
            this->dataGridView1 = (gcnew System::Windows::Forms::DataGridView());
            this->panel3 = (gcnew System::Windows::Forms::Panel());
            this->button3 = (gcnew System::Windows::Forms::Button());
            this->textBox3 = (gcnew System::Windows::Forms::TextBox());
            this->textBox2 = (gcnew System::Windows::Forms::TextBox());
            this->textBox1 = (gcnew System::Windows::Forms::TextBox());
            this->label3 = (gcnew System::Windows::Forms::Label());
            this->label2 = (gcnew System::Windows::Forms::Label());
            this->label1 = (gcnew System::Windows::Forms::Label());
            this->panel2 = (gcnew System::Windows::Forms::Panel());
            this->button2 = (gcnew System::Windows::Forms::Button());
            this->button1 = (gcnew System::Windows::Forms::Button());
            this->saveFileDialog1 = (gcnew
System::Windows::Forms::SaveFileDialog());
            this->saveFileDialog2 = (gcnew
System::Windows::Forms::SaveFileDialog());
            this->panel1->SuspendLayout();
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>dataGridView1))->BeginInit();
            this->panel3->SuspendLayout();
            this->panel2->SuspendLayout();
            this->SuspendLayout();
            //
            // panel1
            //
            this->panel1->Controls->Add(this->dataGridView1);
            this->panel1->Location = System::Drawing::Point(0, 0);
            this->panel1->Name = L"panel1";
            this->panel1->Size = System::Drawing::Size(346, 178);
            this->panel1->TabIndex = 2;
            //
            // dataGridView1
            //
            this->dataGridView1->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;
            this->dataGridView1->Location = System::Drawing::Point(12, 12);
            this->dataGridView1->Name = L"dataGridView1";
            this->dataGridView1->Size = System::Drawing::Size(263, 120);
            this->dataGridView1->TabIndex = 1;
            //
            // panel3
            //
            this->panel3->Controls->Add(this->button3);
            this->panel3->Controls->Add(this->textBox3);
            this->panel3->Controls->Add(this->textBox2);
            this->panel3->Controls->Add(this->textBox1);
            this->panel3->Controls->Add(this->label3);
            this->panel3->Controls->Add(this->label2);
```

```
                    this->panel3->Controls->Add(this->label1);
                    this->panel3->Location = System::Drawing::Point(12, 273);
                    this->panel3->Name = L"panel3";
                    this->panel3->Size = System::Drawing::Size(400, 130);
                    this->panel3->TabIndex = 1;
                    //
                    // button3
                    //
                    this->button3->Location = System::Drawing::Point(103, 91);
                    this->button3->Name = L"button3";
                    this->button3->Size = System::Drawing::Size(75, 23);
                    this->button3->TabIndex = 4;
                    this->button3->Text = L"button3";
                    this->button3->UseVisualStyleBackColor = true;
                    this->button3->Click += gcnew System::EventHandler(this,
&MyForm::button3_Click);
                    //
                    // textBox3
                    //
                    this->textBox3->Location = System::Drawing::Point(163, 65);
                    this->textBox3->Name = L"textBox3";
                    this->textBox3->Size = System::Drawing::Size(146, 20);
                    this->textBox3->TabIndex = 3;
                    //
                    // textBox2
                    //
                    this->textBox2->Location = System::Drawing::Point(163, 39);
                    this->textBox2->Name = L"textBox2";
                    this->textBox2->Size = System::Drawing::Size(146, 20);
                    this->textBox2->TabIndex = 2;
                    //
                    // textBox1
                    //
                    this->textBox1->Location = System::Drawing::Point(163, 13);
                    this->textBox1->Name = L"textBox1";
                    this->textBox1->Size = System::Drawing::Size(146, 20);
                    this->textBox1->TabIndex = 1;
                    //
                    // label3
                    //
                    this->label3->AutoSize = true;
                    this->label3->Location = System::Drawing::Point(22, 64);
                    this->label3->Name = L"label3";
                    this->label3->Size = System::Drawing::Size(35, 13);
                    this->label3->TabIndex = 2;
                    this->label3->Text = L"label3";
                    //
                    // label2
                    //
                    this->label2->AutoSize = true;
                    this->label2->Location = System::Drawing::Point(22, 38);
                    this->label2->Name = L"label2";
                    this->label2->Size = System::Drawing::Size(35, 13);
                    this->label2->TabIndex = 1;
                    this->label2->Text = L"label2";
                    //
                    // label1
                    //
                    this->label1->AutoSize = true;
                    this->label1->Location = System::Drawing::Point(22, 13);
                    this->label1->Name = L"label1";
```

```
                    this->label1->Size = System::Drawing::Size(35, 13);
                    this->label1->TabIndex = 0;
                    this->label1->Text = L"label1";
                    //
                    // panel2
                    //
                    this->panel2->Controls->Add(this->button2);
                    this->panel2->Controls->Add(this->button1);
                    this->panel2->Location = System::Drawing::Point(25, 194);
                    this->panel2->Name = L"panel2";
                    this->panel2->Size = System::Drawing::Size(368, 73);
                    this->panel2->TabIndex = 5;
                    //
                    // button2
                    //
                    this->button2->Location = System::Drawing::Point(163, 48);
                    this->button2->Name = L"button2";
                    this->button2->Size = System::Drawing::Size(75, 23);
                    this->button2->TabIndex = 2;
                    this->button2->Text = L"button2";
                    this->button2->UseVisualStyleBackColor = true;
                    this->button2->Click += gcnew System::EventHandler(this,
&MyForm::button2_Click);
                    //
                    // button1
                    //
                    this->button1->Location = System::Drawing::Point(47, 48);
                    this->button1->Name = L"button1";
                    this->button1->Size = System::Drawing::Size(75, 23);
                    this->button1->TabIndex = 1;
                    this->button1->Text = L"button1";
                    this->button1->UseVisualStyleBackColor = true;
                    this->button1->Click += gcnew System::EventHandler(this,
&MyForm::button1_Click);
                    //
                    // MyForm
                    //
                    this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                    this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                    this->ClientSize = System::Drawing::Size(412, 414);
                    this->Controls->Add(this->panel2);
                    this->Controls->Add(this->panel3);
                    this->Controls->Add(this->panel1);
                    this->Name = L"MyForm";
                    this->Text = L"MyForm";
                    this->Load += gcnew System::EventHandler(this, &MyForm::MyForm_Load);
                    this->panel1->ResumeLayout(false);
                    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>dataGridView1))->EndInit();
                    this->panel3->ResumeLayout(false);
                    this->panel3->PerformLayout();
                    this->panel2->ResumeLayout(false);
                    this->ResumeLayout(false);
                }

#pragma endregion
                DataTable^ Таблиця;  // Об'явлення объекта "Таблиця"
                DataSet^ НабірДаних; // Об'явлення объекта "НабірДаних"
                DataColumn^ Стовпчик1; // Об'явлення объекта Стовпчик1
                DataColumn^ Стовпчик2; // Об'явлення объекта Стовпчик2
```

```
        private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e) {
                // Створюємо екземпляр класу Excel::Application:
                Microsoft::Office::Interop::Excel::Application ^ XL1 = gcnew
Microsoft::Office::Interop::Excel::Application();
                XL1->Visible = true;
                // Задаємо параметр за замовчуванням для його подальшого
                // використання у відповідних методах:
                Object ^ t = Type::Missing;
                // Створюємо нову книгу MS Excel:
                Microsoft::Office::Interop::Excel::Workbook ^ Книга = XL1->Workbooks->Add(t);
                // Об'являємо аркуші в книзі:
                Microsoft::Office::Interop::Excel::Sheets ^ Аркуші = Книга->Worksheets;
                // Вибираємо перший аркуш:
                Microsoft::Office::Interop::Excel::Worksheet ^ Аркуш =
(Microsoft::Office::Interop::Excel::Worksheet ^)Аркуші->Item[1];
                // Якщо хочемо додати ще один аркуш (четвертий) до вже існуючих:
                // _Worksheet ^ Аркуш = safe_cast<_Worksheet^>(Аркуші->Item[ (Object^)1 ]);
                // Записуємо дані у відповідних комірках:
                String ^ colA;
                String ^ colB;
                // Записуємо заголовки стовпчиків на лист Excel
                Аркуш->Range["A1", t]->Value2 = Стовпчик1->Caption;
                Аркуш->Range["B1", t]->Value2 = Стовпчик2->Caption;
                // Записуємо таблицю на лист Excel
                for (int i = 0; i < Таблиця->Rows->Count; i++)
                {
                        colA = "A" + (i+2).ToString();
                        colB = "B" + (i+2).ToString();
                        Аркуш->Range[colA, t]->Value2 = dataGridView1->Rows[i]->Cells[0]-
>Value;
                        Аркуш->Range[colB, t]->Value2 = dataGridView1->Rows[i]->Cells[1]-
>Value;
                }
                // Замовляємо побудову діаграми (графіка) з параметрами за замовчуванням:
                Microsoft::Office::Interop::Excel::Chart ^ Графік =
(Microsoft::Office::Interop::Excel::Chart ^)XL1->Charts->Add(t, t, t, t);
                // Задаємо діапазон значень для побудови графіку:
                Графік->SetSourceData(Аркуш->Range["A2", colB],
Microsoft::Office::Interop::Excel::XlRowCol::xlColumns);
                // Задаємо тип графіка "стовпчикова діаграма" (гістограма):
                Графік->ChartType =
Microsoft::Office::Interop::Excel::XlChartType::xlColumnClustered;
                // Відключаємо легенду графіка:
                Графік->HasLegend = false;
                // Графік має заголовок:
                Графік->HasTitle = true;
                Графік->ChartTitle->Caption = Таблиця->TableName;
                // Підпис осі X:
                Microsoft::Office::Interop::Excel::Axis ^ ГоризонтальнаОсь =
(Microsoft::Office::Interop::Excel::Axis^)Графік->Axes(
                        Microsoft::Office::Interop::Excel::XlAxisType::xlCategory,
Microsoft::Office::Interop::Excel::XlAxisGroup::xlPrimary);
                ГоризонтальнаОсь->HasTitle = true;
                ГоризонтальнаОсь->AxisTitle->Text = Стовпчик1->Caption;
                // Підпис осі Y:
                Microsoft::Office::Interop::Excel::Axis ^ ВертикальнаОсь =
(Microsoft::Office::Interop::Excel::Axis^)Графік->Axes(
                        Microsoft::Office::Interop::Excel::XlAxisType::xlValue,
Microsoft::Office::Interop::Excel::XlAxisGroup::xlPrimary);
                ВертикальнаОсь->HasTitle = true;
                ВертикальнаОсь->AxisTitle->Text = Стовпчик2->Caption;
```

```
            // Збереження графіка в растровому файлі:
            if (MessageBox::Show("Зберегти діаграму?", "Діаграма",
                  MessageBoxButtons::YesNo, MessageBoxIcon::Question) ==
System::Windows::Forms::DialogResult::Yes)
            {
                  if (saveFileDialog2->ShowDialog() ==
System::Windows::Forms::DialogResult::OK)
                  {
                        // Якщо користувач натиснув Зберегти, зберігаємо діаграму
                        XL1->ActiveChart->Export(saveFileDialog2->FileName, t, t);
                  }
            }
      }

      private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
            // Задаємо написи на кнопках
            button3->Text = "Створити таблицю";
            button1->Text = "Створити діаграму Excel";
            button2->Text = "Зберегти таблицю у файл xml";
            // Задаємо заголовок вікна
            Text = "Створення діаграми в MS Excel";
            // Задаємо висоту кнопок
            button1->Height = 25;
            button2->Height = button1->Height;
            button3->Height = button1->Height;
            // Задаємо написи на мітках Lable
            label1->Text = "Назва таблиці";
            label2->Text = "Ім'я 1-го стовпчика";
            label3->Text = "Ім'я 2-го стовпчика";
            // Вирівнюємо кнопки в межах об'єктів Panel
            button1->Dock = DockStyle::Bottom;
            button2->Dock = DockStyle::Bottom;
            button3->Dock = DockStyle::Bottom;
            // Вирівнюємо об'кти Lable та TextBox в межах об'єкту panel3
            label1->Width = label3->Width;
            label2->Width = label3->Width;
            label1->Top = 10; label1->Left = 10;
            label2->Top = (label1->Height + 10) + 10; label2->Left = 10;
            label3->Top = (label1->Height + 10) * 2 + 10; label3->Left = 10;
            textBox1->Top = label1->Top; textBox1->Left = label3->Width + 30;
            textBox2->Top = label2->Top; textBox2->Left = label3->Width + 30;
            textBox3->Top = label3->Top; textBox3->Left = label3->Width + 30;
            textBox1->Width = 110;
            textBox2->Width = textBox1->Width;
            textBox3->Width = textBox1->Width;
            // Задаємо ширину форми з урахуванням ширини об'єктів Lable та TextBox
            Width = 10 + label1->Width + 20 + textBox1->Width + 60;
            // Задаємо висоту panel3 з урахуванням висоти об'єктів Lable, TextBox, button3
            panel3->Height = (label1->Height + 10) * 3 + button3->Height + 30;
            // Задаємо висоту panel2
            panel2->Height = button1->Height * 2;
            // Вирівнюємо об'єкти Panel в межах форми
            panel2->Dock = DockStyle::Bottom;
            panel3->Dock = DockStyle::Bottom;
            dataGridView1->Dock = DockStyle::Fill;
            panel1->Dock = DockStyle::Fill;
            // Властивіст DockStyle::Fill працює не завжди коректно
            //щоб виправити ситуацію можна скористатись "чарівним оператором",
            //який встановить для проблемного об'єкта індекс 0 властивості
            //SetChildIndex батьківського об'єкту
            this->Controls->SetChildIndex(panel1, 0);////////////////////////////////////
```

```
            // Приховуємо панель panel2 з кнопками побудови діаграми та збереження таблиці
            panel2->Visible = false;
            // Встановлюємо фільтр для збереження таблиці у файл XML
            saveFileDialog1->Filter = "Файли XML(*.xml)|*.xml|Всі файли (*.*)|*.*";
            // Встановлюємо фільтр для збереження діаграми у файл JPG
            saveFileDialog2->Filter = "Файли зображень JPG(*.jpg)|*.jpg|Всі файли
(*.*)|*.*";
        }

private: System::Void button2_Click(System::Object^  sender, System::EventArgs^  e) {
        // Зберегти файл tabl.xml:
        if (saveFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK)
        {
            // Якщо користувач натиснув Зберегти, зберігаємо тиблицю
            НабірДаних->WriteXml(saveFileDialog1->FileName);
        }
}

private: System::Void button3_Click(System::Object^  sender, System::EventArgs^  e) {
        Таблиця = gcnew DataTable();
        НабірДаних = gcnew DataSet();
        // Зчитуємо назви стовпчиків з полів textBox
        Стовпчик1 = Таблиця->Columns->Add(textBox2->Text);
        Стовпчик2 = Таблиця->Columns->Add(textBox3->Text);
        // Зчитуємо назву таблиці з поля textBox1
        Таблиця->TableName = textBox1->Text;
        // Пов'язуємо об'єкт dataGridView1 з Таблицею
        dataGridView1->DataSource = Таблиця;
        // Додати об'єкт Таблиця в DataSet
        НабірДаних->Tables->Add(Таблиця);
        // Відображаємо панель з кнопками
        panel2->Visible = true;
}
};
}
```

Завдання до комп'ютерного практикуму №7

Варіанти

1. $\begin{cases} 23{,}82x_1 + 2{,}41x_2 + 3{,}84x_3 = 30{,}25 \\ 2{,}31x_1 + 31{,}48x_2 + 1{,}52x_3 = 41{,}35 \\ 3{,}47x_1 + 4{,}85x_2 + 26{,}18x_3 = 43{,}81 \end{cases}$

2. $\begin{cases} 24{,}18x_1 + 24{,}47x_2 + 3{,}78x_3 = 30{,}74 \\ 2{,}16x_1 + 29{,}85x_2 + 1{,}96x_3 = 40{,}52 \\ 3{,}68x_1 + 4{,}53x_2 + 27{,}14x_3 = 42{,}63 \end{cases}$

3. $\begin{cases} 25{,}14x_1 + 3{,}41x_2 + 4{,}26x_3 = 32{,}48 \\ 3{,}21x_1 + 34{,}56x_2 + 2{,}38x_3 = 45{,}76 \\ 2{,}28x_1 + 4{,}73x_2 + 29{,}15x_3 = 42{,}53 \end{cases}$

4. $\begin{cases} 24{,}35x_1 + 2{,}87x_2 + 3{,}47x_3 = 30{,}54 \\ 2{,}16x_1 + 30{,}62x_2 + 2{,}31x_3 = 40{,}27 \\ 3{,}64x_1 + 4{,}23x_2 + 28{,}14x_3 = 42{,}36 \end{cases}$

5. $\begin{cases} 26{,}13x_1 + 3{,}15x_2 + 4{,}27x_3 = 32{,}46 \\ 3{,}24x_1 + 32{,}04x_2 + 2{,}13x_3 = 43{,}27 \\ 4{,}15x_1 + 5{,}32x_2 + 30{,}81x_3 = 45{,}38 \end{cases}$

6. $\begin{cases} 21{,}65x_1 + 1{,}87x_2 + 3{,}54x_3 = 28{,}13 \\ 1{,}84x_1 + 32{,}51x_2 + 2{,}49x_3 = 40{,}79 \\ 3{,}72x_1 + 4{,}47x_2 + 28{,}35x_3 = 42{,}61 \end{cases}$

7. $\begin{cases} 20{,}93x_1 + 2{,}63x_2 + 3{,}67x_3 = 29{,}85 \\ 2{,}63x_1 + 30{,}74x_2 + 1{,}48x_3 = 40{,}73 \\ 4{,}12x_1 + 5{,}16x_2 + 26{,}35x_3 = 41{,}92 \end{cases}$

8. $\begin{cases} 26{,}15x_1 + 4{,}23x_2 + 5{,}17x_3 = 34{,}28 \\ 4{,}36x_1 + 35{,}18x_2 + 3{,}62x_3 = 45{,}62 \\ 5{,}73x_1 + 6{,}14x_2 + 24{,}36x_3 = 41{,}37 \end{cases}$

9. $\begin{cases} 25{,}28x_1 + 3{,}64x_2 + 4{,}17x_3 = 31{,}25 \\ 2{,}36x_1 + 31{,}15x_2 + 1{,}92x_3 = 40{,}62 \\ 3{,}72x_1 + 4{,}53x_2 + 29{,}31x_3 = 43{,}17 \end{cases}$

10. $\begin{cases} 24{,}17x_1 + 2{,}43x_2 + 3{,}26x_3 = 30{,}42 \\ 2{,}48x_1 + 30{,}14x_2 + 1{,}63x_3 = 40{,}29 \\ 3{,}26x_1 + 4{,}52x_2 + 28{,}47x_3 = 42{,}56 \end{cases}$

11. $\begin{cases} 19{,}82x_1 + 1{,}35x_2 + 4{,}62x_3 = 31{,}42 \\ 2{,}18x_1 + 24{,}67x_2 + 3{,}15x_3 = 42{,}16 \\ 1{,}76x_1 + 3{,}82x_2 + 32{,}14x_3 = 40{,}35 \end{cases}$

12. $\begin{cases} 18{,}73x_1 + 1{,}35x_2 + 2{,}64x_3 = 26{,}53 \\ 3{,}17x_1 + 26{,}43x_2 + 0{,}98x_3 = 30{,}27 \\ 4{,}36x_1 + 2{,}78x_2 + 20{,}31x_3 = 34{,}81 \end{cases}$

13. $\begin{cases} -0{,}12x_1 + 0{,}04x_2 + x_3 = 0{,}903 \\ x_1 - 0{,}417x_2 - 0{,}128x_3 = 0{,}212 \\ -0{,}417x_1 + x_2 + 0{,}046x_3 = 0{,}043 \end{cases}$

14. $\begin{cases} 24{,}61x_1 + 2{,}42x_2 + 3{,}86x_3 = 30{,}24 \\ 2{,}31x_1 + 31{,}48x_2 + 1{,}54x_3 = 41{,}25 \\ 3{,}52x_1 + 4{,}83x_2 + 29{,}12x_3 = 42{,}83 \end{cases}$

15. $\begin{cases} 25{,}72x_1 + 3{,}53x_2 + 4{,}97x_3 = 31{,}35 \\ 3{,}42x_1 + 32{,}59x_2 + 2{,}65x_3 = 41{,}36 \\ 4{,}63x_1 + 4{,}94x_2 + 30{,}23x_3 = 43{,}94 \end{cases}$

16. $\begin{cases} 23{,}92x_1 + 2{,}51x_2 + 4{,}03x_3 = 30{,}24 \\ 2{,}61x_1 + 31{,}78x_2 + 1{,}82x_3 = 41{,}65 \\ 3{,}57x_1 + 4{,}95x_2 + 26{,}78x_3 = 43{,}82 \end{cases}$

17. $\begin{cases} 25{,}31x_1 + 3{,}16x_2 + 4{,}28x_3 = 32{,}45 \\ 2{,}38x_1 + 32{,}14x_2 + 1{,}84x_3 = 40{,}57 \\ 3{,}52x_1 + 5{,}16x_2 + 31{,}24x_3 = 43{,}18 \end{cases}$

18. $\begin{cases} 23{,}24x_1 + 2{,}41x_2 + 3{,}43x_3 = 31{,}52 \\ 2{,}17x_1 + 32{,}18x_2 + 1{,}15x_3 = 40{,}84 \\ 3{,}56x_1 + 4{,}29x_2 + 30{,}42x_3 = 42{,}73 \end{cases}$

19. $\begin{cases} 25{,}21x_1 + 3{,}14x_2 + 4{,}21x_3 = 32{,}35 \\ 2{,}28x_1 + 32{,}24x_2 + 2{,}34x_3 = 40{,}73 \\ 3{,}61x_1 + 4{,}16x_2 + 30{,}18x_3 = 42{,}39 \end{cases}$

20. $\begin{cases} 24{,}86x_1 + 2{,}51x_2 + 3{,}82x_3 = 30{,}94 \\ 2{,}19x_1 + 29{,}87x_2 + 1{,}76x_3 = 40{,}56 \\ 3{,}61x_1 + 4{,}46x_2 + 29{,}18x_3 = 43{,}27 \end{cases}$

21. $\begin{cases} 22{,}31x_1 + 1{,}12x_2 + 2{,}56x_3 = 30{,}23 \\ 1{,}94x_1 + 30{,}27x_2 + 1{,}34x_3 = 40{,}15 \\ 2{,}39x_1 + 3{,}45x_2 + 29{,}61x_3 = 42{,}36 \end{cases}$

22. $\begin{cases} 23{,}48x_1 + 2{,}62x_2 + 3{,}56x_3 = 30{,}65 \\ 2{,}51x_1 + 32{,}17x_2 + 1{,}62x_3 = 41{,}12 \\ 3{,}67x_1 + 4{,}53x_2 + 25{,}81x_3 = 43{,}95 \end{cases}$

23. $\begin{cases} 23{,}51x_1 + 1{,}32x_2 + 2{,}76x_3 = 29{,}14 \\ 1{,}21x_1 + 30{,}38x_2 + 0{,}44x_3 = 39{,}15 \\ 2{,}42x_1 + 3{,}73x_2 + 28{,}02x_3 = 41{,}73 \end{cases}$

24. $\begin{cases} 24{,}31x_1 + 2{,}12x_2 + 3{,}56x_3 = 30{,}15 \\ 2{,}11x_1 + 31{,}18x_2 + 1{,}24x_3 = 40{,}23 \\ 3{,}42x_1 + 4{,}53x_2 + 29{,}42x_3 = 42{,}53 \end{cases}$

25. $\begin{cases} 22{,}68x_1 + 1{,}92x_2 + 2{,}36x_3 = 30{,}57 \\ 2{,}23x_1 + 30{,}56x_2 + 1{,}84x_3 = 40{,}81 \\ 3{,}17x_1 + 4{,}13x_2 + 28{,}95x_3 = 4315 \end{cases}$

26. $\begin{cases} 24{,}25x_1 + 2{,}86x_2 + 3{,}37x_3 = 30{,}44 \\ 2{,}06x_1 + 30{,}52x_2 + 2{,}31x_3 = 40{,}36 \\ 3{,}67x_1 + 4{,}26x_2 + 29{,}14x_3 = 42{,}63 \end{cases}$

27. $\begin{cases} 24{,}65x_1 + 2{,}38x_2 + 3{,}42x_3 = 31{,}45 \\ 2{,}38x_1 + 31{,}57x_2 + 1{,}64x_3 = 42{,}36 \\ 3{,}27x_1 + 4{,}13x_2 + 28{,}45x_3 = 45{,}28 \end{cases}$

28. $\begin{cases} 24{,}65x_1 + 2{,}46x_2 + 3{,}91x_3 = 30{,}28 \\ 2{,}35x_1 + 31{,}52x_2 + 1{,}58x_3 = 40{,}29 \\ 3{,}55x_1 + 4{,}87x_2 + 29{,}16x_3 = 42{,}87 \end{cases}$

29. $\begin{cases} 23{,}24x_1 + 2{,}11x_2 + 3{,}94x_3 = 30{,}61 \\ 2{,}23x_1 + 32{,}17x_2 + 2{,}04x_3 = 41{,}45 \\ 3{,}37x_1 + 4{,}13x_2 + 28{,}19x_3 = 43{,}64 \end{cases}$

30. $\begin{cases} 21{,}83x_1 + 3{,}53x_2 + 4{,}57x_3 = 30{,}75 \\ 3{,}42x_1 + 31{,}64x_2 + 2{,}38x_3 = 41{,}63 \\ 5{,}02x_1 + 6{,}12x_2 + 28{,}25x_3 = 42{,}81 \end{cases}$

Контрольні питання

1) Яку бібліотеку потрібно підключити для використання функцій MS Word в програмі?
2) Яку бібліотеку потрібно підключити для використання функцій MS Excel в програмі?
3) Як підключити зовнішню бібліотеку до проекту?
4) Яка функція дозволяє перевірити орфографії засобами MS Word?
5) Як можна записати значення в комірки MS Excel?
6) Яка функція відповідає за вибір типу діаграму MS Excel?
7) Як приховати від користувача використання можливостей MS Excel в програмі?
8) Як організувати збереження у обраному користувачем файлі XML заданий у об'єкті DataSet набір даних?
9) Як додати в програму можливість використання бібліотеки функцій Microsoft VisualBasic?

Рекомендована література

Базова

1) Хортон, Айвор. Visual C++ 2010: полный курс.: Пер. с англ.- М.: ООО «И.Д. Вильямс», 2011. – 1216 с.
2) Зиборов В.В. MS Visual C ++2010 в среде .NET. Библиотека программиста – СПб.: Питер, 2012. – 320 с.
3) Пахомов Б.И. C/C++ и MS Visual C++ 2010 для начинающих. – СПб.: БХВ-Петербург, 2011. – 736 с.
4) Стенли Б. Липпман, Жози Лажойе, Барбара Э. Му Язык программирования C++. Базовый курс. М.: Вильямс, 2014. – 1120 с.
5) Страуструп, Бьярне. Программирование: принципы и практика использования C++, 2-е изд. - М.: ООО «И.Д. Вильямс», 2016. – 1328 с.
6) Прата С. Язык программирования C++. Лекции и упражнения. Учебник. - СПб.: ООО «ДиаСофтЮП», 2005. - 1104 с.
7) Мейерс С. Эффективное использование C++. 50 рекомендаций по улучшению ваших программ и проектов. - М.: Питер-ДМК, 2006. – 240 с.

Допоміжна

6) Аверкин В.П., Бобровский А.И. и др. под ред. Хомоненко А.Д. Программирование на C++. Учебное пособие.Корона-Принт, 1999. – 252 с.
7) Александреску, Андрей. Современное проектирование на C++. Серия C++ In-Depth, т.3.: Пер. с англ. – М.: Вильямс, 2002. – 336 с.
8) Астахова, И.Ф., Власов, С.В. Язык C++. Учебное пособие. И.Ф. Астахова, С.В. Власов, В.В. Фертиков, А.В. Ларин. – Мн.: Новое знание, 2003. – 203 с.
9) Седжвик, Роберт. Фундаментальные алгоритмы на C++. Анализ/Структуры данных/Сортировка/Поиск: Пер. с англ./ Роберт Седжвик. – К.: ДиаСофт, 2001. – 688 с.
10) Седжвик, Роберт. Фундаментальные алгоритмы на C++. Алгоритмы на графах: Пер. с англ./ Роберт Седжвик. – СПб.: ДиаСофтЮП, 2002. – 496 с.
11) Дейтел Х.М., Дейтел П.Дж. Как программировать на C++: Пятое издание. Пер. с англ. – М.: ООО «Бином-Пресс», 2008. - 1456 с.

Інформаційні ресурси

12) Язык программирования C++ [Електрон. ресурс] // Основы программирования на языках Си и C++ для начинающих - Режим доступу: http://cppstudio.com/cat/274/

13) Бендюг Владислав Іванович [Електрон. ресурс] // Офіційний сайт – Режим доступу: http://бендюг.укр/
14) Єдине інформаційне середовище - Національний технічний університет України «КПІ ім. І.Сікорського» [Електрон. ресурс] // Електронний КАМПУС НТУУ «КПІ» – Режим доступу: http://login.kpi.ua/

Додаток А

Формули для довідок

1. Рівняння Менделєєва-Клапейрона:

$$P \cdot V = \frac{m}{M} \cdot R \cdot T,$$

де *P* – тиск; *V* – об'єм; *m* – маса газу; *M* – молекулярна маса; *R* – універсальна газова стала; *T* – абсолютна температура.

2. Співвідношення між елементами трикутника зі сторонами a, b, c та протилежними кутами A, B, C.

2.1. Теорема синусів:

$$\frac{a}{\sin A} = \frac{b}{\sin B} = \frac{c}{\sin C}$$

де R - радіус описаного кола.

2.2. Теорема косинусів:

$$a^2 = b^2 + c^2 - 2bc \cos A.$$

2.3. Площа трикутника:

$$S = \frac{a\, h_a}{2}$$

де h_a висота трикутника.

2.4. Форма Герона:

$$S = \sqrt{p\,(p-a)(p-b)(p-c)}, \qquad \left(p = \frac{a+b+c}{2}\right).$$

2.5. $$S = \frac{1}{2}\, a\, b \sin c.$$

2.6. $$S = r \cdot p,$$

$r -$ радіус кола,вписаного у трикутнок.

2.7. Для рівнобічного трикутника:

$$S = \frac{a^2 \sqrt{3}}{4}.$$

3. Площа фігур.

3.1. Паралелограма:

$$S = b\, h,$$

де b – сторона, h - висота.

3.2. Ромба:

$$S = \frac{1}{2}\, d_1\, d_2$$

де d_1 та d_2 – діагоналі.

3.3. Трапеції:

$$S = \frac{a+b}{2}\, h,$$

де a та b – сторони, h - висота.

$$S = m\, h,$$

де - m – середня лінія.

3.4. Квадрата:

$$S = a^2,$$

де a – сторона.

3.5. Прямокутника:

$$S = a\, b,$$

де a та b – сторони.

3.6. Кола:

$$S = \pi\, R^2,$$

де R – радіус.

4. Об'єми різних фігур:

де S – площа основи, H –висота.

4.1. Призми:

$$V = S\, H.$$

4.2. Циліндра:

$$V = S\, H.$$

4.3. Піраміди:

$$V = \frac{1}{3}\acute{S} \cdot H$$

4.4. Конуса:

$$V = \frac{1}{3} \cdot \acute{S} \cdot H,$$

де

$$\acute{S} = \pi\, R^2$$

4.5 Кулі:

$$V = \frac{4}{3}\, \pi\, R^3$$

де R – радіус.

5. Квадратні рівняння:

5.1.

$$a\, x^2 + b\, x + c = 0;$$

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4\, a\, c}}{2\, a}.$$

5.2. $$x^2 + p\,x + q = 0;$$
$$x_{1,2} = -\frac{p}{2} + \sqrt{\frac{p^2}{4} - q}.$$

5.3. Теорема Віета:
коли $x^2 + p \cdot x + q = 0$, то $x_1 + x_2 = -p$, $x_1 \cdot x_2 = q$.

6. Рішення системи рівнянь з двома невідомими:

Для $\begin{cases} a_1 \cdot x + b_1 \cdot y = c_1, \\ a_2 \cdot x + b_2 \cdot y = c_2, \end{cases}$

$\Delta = a_1 \cdot b_2 - a_2 \cdot b_1$;
$\Delta_x = c_1 \cdot b_2 - c_2 \cdot b_1$;
$\Delta_y = a_1 \cdot c_2 - a_2 \cdot c_1$.

Тоді $x = \frac{\Delta_x}{\Delta}$; $y = \frac{\Delta_y}{\Delta}$

7. Довжина відрізка АВ $(A(x_1,\ y_1), B(x_2,\ y_2))$,
$$l_{AB} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}.$$

8. Яущо точка $D(x,\ y)$ знаходиться у площі трикутника АВС, де $A(x_1,\ y_1), B(x_2,\ y_2), C(x_3,\ y_3)$, то
$$S_{ABC} = S_{ADC} + S_{CDB} + S_{ADB}.$$